



City Research Online

City St George's, University of London

Citation: Kosari, A. (2025). Cyber-Resilience of Connected and Autonomous Vehicles (CAVs). (Unpublished Doctoral thesis, City St George's, University of London)

This is the accepted version of the paper.

This version of the publication may differ from the final published version. To cite this item please consult the publisher's version.

Permanent repository link: <https://openaccess.city.ac.uk/id/eprint/37023/>

Copyright and Reuse: Copyright and Moral Rights remain with the author(s) and/or copyright holders. Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge, unless otherwise indicated, provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way. For full details of reuse please refer to [City Research Online policy](#).

Cyber-Resilience of Connected and Autonomous Vehicles (CAVs)

Aria Kosari

Submitted for Doctor of Philosophy in Computer Science

**School of Science and Technology
City St.George's University
of London
United Kingdom**

March 2025

Contents

1	Introduction	13
1.1	Connected Places Catapult	13
1.2	Connected & Autonomous Vehicles	13
1.3	Problem Statement	17
2	Background & Literature Review	19
2.1	In-Vehicle Network	19
2.1.1	Electronic Control Units	19
2.1.2	Controller Area Network (CAN)	19
2.2	Sensors	21
2.2.1	Light Detection and Ranging (LiDAR)	22
2.2.2	Radio Detection and Ranging (RADAR)	22
2.2.3	Camera	22
2.2.4	Global Positioning Systems (GPS)	22
2.3	Architecture	23
2.4	Safety	25
2.5	Vehicular Communications	31
2.5.1	Vehicle-to-Vehicle (V2V)	31
2.5.2	Vehicle-to-Infrastructure (V2I)	31
2.5.3	Vehicle-to-Everything (V2X)	31
2.6	Vulnerabilities	32
2.6.1	Attack Surface	32
2.6.2	Attacks	33
2.7	Defences	36
2.7.1	Authentication	37
2.7.2	Intrusion Detection Systems (IDS)	37
2.7.3	Intrusion Tolerance	38
2.8	Vehicle Standards	42
2.9	Modelling	47
2.9.1	Stochastic Activity Network (SAN)	47
2.9.2	ADVISE	47
2.9.3	GSPN	50
2.9.4	PRISM	50
2.9.5	Other	52
3	Research Methodology	54

3.1	Key Objectives	54
3.2	Methodology	54
3.2.1	The Möbius Tool.....	54
3.2.2	Vehicle Simulators.....	55
3.2.3	Hazard Models	56
3.2.4	Attack Models	56
3.2.5	Model Validation	57
4	Concept of Modelling	58
4.1	SAN.....	59
4.2	Reward Model.....	62
4.3	Attack Model.....	63
4.4	DESSERT'2022	64
4.5	Combination Testing – Improved Model	64
4.5.1	Main Model.....	64
4.5.1.1	Discussion.....	71
4.5.2	Realistic Attack.....	72
4.5.2.1	Discussion.....	74
4.5.3	Sensitivity Analysis	76
4.6	Partitioned Model.....	77
4.6.1	Main Model.....	80
4.6.1.1	Discussion.....	84
4.6.2	Realistic Attack.....	84
4.6.2.1	Discussion.....	85
4.6.3	Sensitivity Analysis	86
5	Uncertainty in Model Parameters	87
5.1	The Lomax Distribution	87
5.1.1	Attack Models.....	88
5.2	Lomax vs. Exponential.....	90
5.2.1	Results.....	90
5.2.1.1	Discussion.....	94
6	Defences	95
6.1	Defence-in-Depth	95
6.1.1	Main Model.....	97
6.1.1.1	Discussion.....	100
6.1.2	Realistic Attack.....	101

6.1.2.1	Discussion.....	102
6.1.3	Weak vs. Strong Adversary	103
6.1.3.1	Discussion.....	104
6.1.4	Sensitivity Analysis	105
6.2	Intrusion Tolerance	108
6.2.1	Composed Models	109
6.2.2	Attacks	110
6.2.3	Intrusion Tolerant Components	111
6.2.4	Fixed Model.....	115
6.2.4.1	Discussion.....	118
6.2.5	Dynamic Model (Replicas).....	119
6.2.5.1	Discussion.....	123
6.2.6	Dynamic Model (Replicas & Attacks).....	124
6.2.6.1	Discussion.....	127
6.2.7	Dynamic Model (Replicas, Attacks & Cleansing).....	128
6.2.7.1	Discussion.....	131
6.2.8	Sensitivity Analysis	132
7	Model Validation	134
7.1	Initial Models	134
7.2	Partitioned Model.....	135
7.3	Uncertainty in Models.....	136
7.3.1	Lomax vs. Exponential (Time Intervals)	136
7.4	Defence-in-Depth	137
7.4.1	Baseline Comparison	137
7.4.2	Independence of Layers (Attacks)	137
7.5	Intrusion Tolerance	139
8	Discussions and Lessons Learned	140
9	Conclusions	143
10	Further Work	146
11	References.....	148
12	Appendix 1.....	157
12.1	Types of Modelling	157
12.2	Background & Literature Review (Extended).....	163
13	Appendix 2.....	215
13.1	Separate Vulnerability Modes Model (Improved DESSERT'2022).....	215

13.2	Combined Vulnerability Modes Model.....	222
13.3	Partitioned Vehicle Architecture Model.....	229
13.4	Defence-in-Depth Model.....	236
13.5	Uncertainty Model (Lomax and Exponential Distribution)	256
13.6	Exponential Time Intervals Model.....	269
13.7	Exponential Time Intervals Model (Deterministic).....	282
13.8	Lomax Time Intervals Model.....	294
13.9	Intrusion Tolerance Models.....	308
14	Appendix 3.....	352
14.1	Vulnerability Mode Graphs.....	352
14.2	Partitioned Model Graphs.....	354
14.3	Defence-in-Depth Graphs.....	355

List of Tables and Figures

Table 1: CAV – Levels of Autonomy (SAE, 2021).....	14
Table 2: Vehicular Communications (El-Rewini et al., 2020)	15
Table 3: Vehicle Sensors (Malik and Sun, 2020)	16
Table 4: Vehicular Vulnerabilities (Dibaei et al., 2019; El-Rewini et al., 2020)	35
Table 5: Table of Fixed Parameters	62
Table 6: Accident Reward Model	63
Table 7: Attack Model.....	64
Table 8: CAV Model Parameters (Improved)	65
Table 9: CAV Mode Combinations	66
Table 10: Single Vulnerability Mode Data	67
Table 11: Two Vulnerability Mode Data.....	68
Table 12: Three Vulnerability Mode Data.....	69
Table 13: Four Vulnerability Mode Data.....	70
Table 14: Realistic Attack Parameters – Vulnerability Modes.....	72
Table 15: Realistic Risk of Accident - 1 Mode (Raw Data)	72
Table 16: Realistic Risk of Accident - 2 Modes (Raw Data)	73
Table 17: Realistic Risk of Accident - 3 Modes (Raw Data)	73
Table 18: Realistic Risk of Accident - 4 Modes (Raw Data)	74
Table 19: Sensitivity Analysis Parameters – Vulnerability Modes	76
Table 20: False Hazard Sensitivity Data (All Modes)	76
Table 21: Attack Rate Sensitivity Data (All Modes)	76
Table 22: Attack Reset Sensitivity Data (All Modes)	77
Table 23: Attack Success Sensitivity Data (All Modes).....	77
Table 24: Partition Vulnerability Mode Equivalents	80
Table 25: Model Parameters of the System with Partitions.....	81
Table 26: List of Partition Experiments.....	81
Table 27: Individual Compromised Partition Data.....	82
Table 28: Two Compromised Partitions Data	83
Table 29: Raw Data - Realistic Attack (Single Partitioned)	84
Table 30: Raw Data - Realistic Attack (Combined Partitions).....	85
Table 31: Sensitivity Analysis Parameters – Partitioned Model	86
Table 32: Attack Reset Sensitivity - Raw Data (All Partitions)	86
Table 33: Lomax Attack Parameters.....	88
Table 34: Lomax Distribution Attack Model.....	89
Table 35: Exponential Distribution Attack Model.....	89
Table 36: Lomax vs. Exponential - Raw Data (Baseline)	90
Table 37: Lomax vs. Exponential - Raw Data - Confidence Levels (Baseline)	90
Table 38: Lomax vs. Exponential - Raw Data (Hazard Perception).....	90
Table 39: Lomax vs. Exponential - Raw Data - Confidence Intervals (Hazard Perception) ...	90
Table 40: Lomax vs. Exponential - Raw Data (Overlooked Hazard).....	91
Table 41: Lomax vs. Exponential - Raw Data - Confidence Levels (Overlooked Hazard)	91
Table 42: Lomax vs. Exponential - Raw Data (Safety Monitor).....	91
Table 43: Lomax vs. Exponential - Raw Data - Confidence Levels (Safety Monitor)	91
Table 44: Lomax vs. Exponential - Raw Data (Hazard + Overlooked)	92
Table 45: Lomax vs. Exponential - Raw Data - Confidence Levels (Hazard + Overlooked) .	92

Table 46: Lomax vs. Exponential - Raw Data (Hazard + Safety Monitor)	92
Table 47: Lomax vs. Exponential - Raw Data - Confidence Levels (Hazard + Safety Monitor)	92
Table 48: Lomax vs. Exponential - Raw Data (Overlooked + Safety Monitor).....	93
Table 49: Lomax vs. Exponential - Raw Data - Confidence Levels (Overlooked + Safety Monitor)	93
Table 50: Lomax vs. Exponential - Raw Data (All Partitions).....	93
Table 51: Lomax vs. Exponential - Raw Data - Confidence Levels (All Partitions)	94
Table 52: Defence-in-Depth Parameters.....	97
Table 53: Defence-in-Depth - Raw Data (1 Partition).....	98
Table 54: Defence-in-Depth - Raw Data (2 Partitions)	99
Table 55: Defence-in-Depth 'Realistic' Parameters	101
Table 56: Defence-in-Depth - 'Realistic' - Raw Data (1 Partition)	101
Table 57: Defence-in-Depth - 'Realistic' - Raw Data (2 Partitions)	102
Table 58: Defence-in-Depth - Weak & Strong Attack Parameters.....	103
Table 59: Weak vs. Strong - All Partitions	103
Table 60: Weak vs. Strong - Hazard Perception.....	103
Table 61: Weak vs. Strong - Hazard + Overlooked.....	103
Table 62: Weak vs. Strong - Hazard + Safety Monitor	103
Table 63: Weak vs. Strong – Overlooked Hazard	104
Table 64: Weak vs. Strong - Overlooked + Safety Monitor	104
Table 65: Weak vs. Strong - Safety Monitor	104
Table 66: Sensitivity Analysis Parameters - Defence-in-Depth	105
Table 67: Defence-in-Depth - Attack Reset Sensitivity (Raw Data).....	105
Table 68: Defence-in-Depth - Hazard Cleansing Sensitivity (Raw Data).....	106
Table 69: Defence-in-Depth - Overlooked Cleansing Sensitivity (Raw Data).....	106
Table 70: Defence-in-Depth - Safety Cleansing Sensitivity (Raw Data)	106
Table 71: Defence-in-Depth - Hazard Success Sensitivity (Raw Data)	106
Table 72: Defence-in-Depth - Overlooked Success Sensitivity (Raw Data).....	107
Table 73: Defence-in-Depth - Safety Success Sensitivity (Raw Data).....	107
Table 74: Intrusion Tolerance - Attack Combination Table.....	108
Table 75: Intrusion Tolerance Parameters - Static Model	116
Table 76: Intrusion Tolerance - Static (Raw Data - 1 Partition).....	116
Table 77: Intrusion Tolerance - Static (Confidence Levels - 1 Partition).....	116
Table 78: Intrusion Tolerance - Static (Raw Data - 2 Partitions)	117
Table 79: Intrusion Tolerance - Static (Confidence Levels - 2 Partitions)	118
Table 80: Intrusion Tolerance Parameters - Dynamic Replicas	120
Table 81: Intrusion Tolerance - Dynamic Replicas (Raw Data - 1 Partition)	121
Table 82: Intrusion Tolerance - Dynamic Replicas (Confidence Levels - 1 Partition)	121
Table 83: Intrusion Tolerance - Dynamic Replicas (Raw Data - 2 Partitions).....	122
Table 84: Intrusion Tolerance - Dynamic Replicas (Confidence Levels - 2 Partitions).....	122
Table 85: Intrusion Tolerance Parameters - Dynamic Replicas & Adversary.....	124
Table 86: Intrusion Tolerance - Dynamic Replicas & Attacks (Raw Data – 1 Partition)	125
Table 87: Intrusion Tolerance - Dynamic Replicas & Attacks (Confidence Levels - 1 Partition)	125
Table 88: Intrusion Tolerance - Dynamic Replicas & Attacks (Raw Data - 2 Partitions).....	126

Table 89: Intrusion Tolerance - Dynamic Replicas & Attacks (Confidence Levels - 2 Partitions).....	126
Table 90: Intrusion Tolerance Parameters - Dynamic Replicas, Adversary and Cleansing..	128
Table 91: Intrusion Tolerance - Dynamic Replicas, Attacks & Cleansing (Raw Data - 1 Partition)	129
Table 92: Intrusion Tolerance - Dynamic Replicas, Attacks & Cleansing (Confidence Levels - 1 Partition).....	129
Table 93: Intrusion Tolerance - Dynamic Replicas, Attacks & Cleansing (Raw Data - 2 Partitions).....	130
Table 94: Intrusion Tolerance - Dynamic Replicas, Attacks & Cleansing (Confidence Levels - 2 Partitions).....	130
Table 95: Intrusion Tolerance Sensitivity Analysis Parameters	132
Table 96: Intrusion Tolerance Sensitivity Analysis - Replica Cleansing (Raw Data).....	133
Table 97: Intrusion Tolerance Sensitivity Analysis - Replica Cleansing (Confidence Levels)	133
Table 98: Intrusion Tolerance Sensitivity Analysis - Replicas (Raw Data)	133
Table 99: Intrusion Tolerance Sensitivity Analysis - Replicas (Confidence Levels).....	133
Table 100: DESSERT vs. Multi-Mode Validation (Baseline).....	134
Table 101: DESSERT vs. Multi-Mode Validation (AccidentScale)	134
Table 102: DESSERT vs. Multi-Mode Validation (DelayScale)	134
Table 103: DESSERT vs. Multi-Mode Validation (HazardScale)	134
Table 104: DESSERT vs. Multi-Mode Validation (LateCH2AccScale).....	134
Table 105: DESSERT vs. Multi-Mode Validation (OH2CH_scale)	134
Table 106: DESSERT vs. Multi-Mode Validation (All Vulnerability Modes).....	134
Table 107: Architecture vs. Modes (Baseline)	135
Table 108: Architecture vs. Modes (Hazard Perception).....	135
Table 109: Architecture vs. Modes (Hazard + Overlooked).....	135
Table 110: Architecture vs. Modes (Hazard + Safety Monitor)	135
Table 111: Architecture vs. Modes (Overlooked Hazards)	135
Table 112: Architecture vs. Modes (Overlooked + Safety Monitor).....	135
Table 113: Architecture vs. Modes (Safety Monitor).....	135
Table 114: Architecture vs. Modes (All)	135
Table 115: Lomax vs. Exponential Validation	136
Table 116: Defence-in-Depth Validation (Baseline Comparison).....	137
Table 117: Defence-in-Depth - Conditional Probability - Success Probability Swap.....	138
Table 118: Conditional Probability - Hazard Perception.....	138
Table 119: Conditional Probability - Overlooked Hazards	138
Table 120: Conditional Probability - Safety Monitor	138
Table 121: Intrusion Tolerance Validation - Baselines (Raw Data).....	139
Table 122: Intrusion Tolerance Validation - Baselines (Confidence Levels).....	139
Figure 1: General CAN Frame Structure (Aliwa et al., 2020).....	20
Figure 2: Gateway E/E Architecture (Dibaei et al., 2019).....	23
Figure 3: Domain E/E Architecture (Dibaei et al., 2019)	24
Figure 4: Centralized E/E Architecture (Dibaei et al., 2019)	25
Figure 5: Vehicle Attack Surface (Malik and Sun, 2020)	32

Figure 6: Hazard Perception State Machine	59
Figure 7: CAV Attack Model	60
Figure 8: Hazard and Attack Model Join	61
Figure 9: Single Vulnerability Mode Attack	66
Figure 10: Two Vulnerability Modes Attacked	67
Figure 11: Three Vulnerability Modes Attacked	69
Figure 12: Four Vulnerability Modes Attacked	70
Figure 13: CAV Partition Block Diagram	78
Figure 14: Composed Model with Partitions	78
Figure 15: Hazard Partition	78
Figure 16: Overlooked Hazard Partition	79
Figure 17: Safety Monitor Partition	79
Figure 18: Single Partition Attacks	82
Figure 19: Combined Partitions Attacked	83
Figure 20: Hazard Perception - Defence-in-Depth	95
Figure 21: Overlooked Hazard - Defence-in-Depth	96
Figure 22: Safety Monitor - Defence-in-Depth	96
Figure 23: Defence-in-Depth (1 Partition)	98
Figure 24: Defence-in-Depth (2 Partitions)	99
Figure 25: Fixed Intrusion Tolerance	109
Figure 26: Dynamic Intrusion Tolerance	110
Figure 27: Intrusion Tolerance Attack Model	111
Figure 28: Intrusion Tolerance Attack Model - Dynamic Attack Rates	111
Figure 29: Intrusion Tolerance Mechanism – Perception	112
Figure 30: Dynamic Intrusion Tolerance Mechanism – Overlooked Hazard	113
Figure 31: Dynamic Intrusion Tolerance Mechanism – Overlooked Hazard (Dynamic Cleansing)	114
Figure 32: Intrusion Tolerance - Static Model (1 Partition)	116
Figure 33: Intrusion Tolerance - Static Model (2 Partitions)	117
Figure 34: Intrusion Tolerance - Dynamic Replicas (1 Partition)	121
Figure 35: Intrusion Tolerance - Dynamic Replicas (2 Partitions)	122
Figure 36: Intrusion Tolerance - Dynamic Replicas & Attacks (1 Partition)	125
Figure 37: Intrusion Tolerance - Dynamic Replicas & Attacks (2 Partitions)	126
Figure 38: Intrusion Tolerance - Dynamic Replicas, Attacks & Cleansing (1 Partition)	129
Figure 39: Intrusion Tolerance - Dynamic Replicas, Attacks & Cleansing (2 Partitions)	130
Figure 40: Lomax vs. Exponential Subplots (First Group)	136
Figure 41: Lomax vs. Exponential Subplots (Second Group)	137

List of Abbreviations

Acronym	Meaning
ADVISE	Adversary View Security Evaluation
AE	Adversarial Example
AI	Artificial Intelligence
ANN	Artificial Neural Network
AV	Autonomous Vehicle
BFT	Byzantine Fault Tolerance
BSI	British Standards Institution
CAN	Controller Area Network
CAPEC	Common Attack Pattern Enumerations and Classifications
CARLA	Car Learning to Act
CAV	Connected and Autonomous Vehicle
CHASSIS	Combined Harm Assessment of Safety and Security for Information Systems
CPC	Connected Places Catapult
CSMA	Carrier Sensitive Multiple Access
CVSS	Common Vulnerability Scoring System
DDoS	Distributed Denial-of-Service
DL	Deep Learning
DNN	Deep Neural Network
DoS	Denial-of-Service
DREAD	Damage, Reproducibility, Exploitability, Affected Users, Discoverability
ECU	Electronic Control Unit
EIT	European Institute of Technology
EVITA	E-Safety Vehicle Intrusion Protected Applications
FGSM	Fast Gradient Sign Method
FMEA	Failure Mode and Effect Analysis
FMVEA	Failure Mode, Effect and Vulnerability Analysis
GPS	Global Positioning System
GSPN	Generalized Stochastic Petri Net
HAZOP	Hazard and Operability
HEAVENS	Healing Vulnerabilities to Enhance Software Security & Safety
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronic Engineers
IET	Institute of Engineering and Technology
IGSM	Iterative Gradient Sign Method
ISO	International Standards Organization
LiDAR	Light Detection and Ranging
LIN	Local Interconnect Network
MDP	Markov Decision Process
MitM	Man-in-the-Middle
ML	Machine Learning
MOST	Media Oriented Systems Transport
OBU	On-Board Unit
OCTAVE	Operationally Critical Threat, Asset and Vulnerability Evaluation
P2P	Peer-to-Peer
PASTA	Process for Attack Simulation and Threat Analysis
PIT	Proactive Intrusion Tolerance
PRISM	Probabilistic Symbolic Model
PTA	Probabilistic Timed Automata
Radar	Radio Detection and Ranging
RNN	Recurrent Neural Network
RSS	Responsibility-Sensitive-Safety
RSU	Roadside Unit
SAE	Society of Automotive Engineers
SAN	Stochastic Activity Network
SAN	Stochastic Activity Network
STRIDE	Spoofing, Tampering, Repudiation, Information Disclosure, Denial-of-Service, Elevation of Privilege
SVM	Support Vector Machine
TAL	Threat Agent Library
TCU	Telematics Control Unit
TDMA	Time Division Multiple Access
V2I	Vehicle-to-Infrastructure
V2V	Vehicle-to-Vehicle
V2X	Vehicle-to-Everything
VANET	Vehicular Ad-hoc Network

Acknowledgements

I would like to thank my supervisors, Dr. Peter Popov, and Professor Rajkumar Roy for their help throughout a particularly tricky subject of research, as well as their patience as I attempted to overcome the hurdles that come with higher level university research, as well as their continued support. I am particularly grateful to Connected Places Catapult (CPC) and the European Institute of Technology for their backing both financially and academically, as I have learned a lot from my time in contact with both. I will be forever grateful to my parents for being at my side throughout the very difficult times that have happened throughout the years and for helping me reach the many milestones in my education and personal life as I am certain I would have stopped much earlier had I been standing alone throughout the years.

Abstract

In recent years, intelligent transport systems have become an area of interest, as they can make transport safer and more convenient for passengers. This has been attempted in public transport before, but never in private forms of transport such as cars, which are far less predictable, requiring the user to relinquish existing control over their vehicle and its decision making to another, non-human system.

It is known that vehicle accidents are very costly and happen frequently, with human error being one of the key reasons behind this. Therefore, a better decision-making system that does not make such error is of great interest, which is where the concept of Connected and Autonomous Vehicles (CAVs) comes into play.

Currently, there are either connected vehicles or autonomous vehicles, which are vehicles that communicate with one another and ones that can make their own decisions respectively, but not both. CAVs aim to combine these, using the connected aspect to increase information about surroundings (i.e., obstacles, traffic, and vehicle manoeuvres) in combination with the autonomous features to allow a vehicle to make accurate, safe, and informed decisions.

However, the systems of CAVs are not perfect, being susceptible to faults and attacks which can have severe safety implications. Many of these attacks are done by exploiting the newfound connectivity of the vehicle, through which they can manipulate information being sent to the vehicle to affect the decision making of the autonomous component, sometimes even controlling the vehicle itself.

To assess how well a vehicle holds up against attacks, I must look at cyber-resilience, which is what I aim to assess in this project, with proper, realistic models and results to back up any arguments made regarding resilience. Stochastic models of the vehicle operation under attacks will be created in the Mobius software, looking at how various parts of the system behave under attack and in the presence of different defence mechanisms.

For this research, the focus of attacks will be on those affecting the safety of the vehicle, i.e., attacks which disrupt the function of the vehicle significantly, which lead to accidents that would not have happened without attacks. By the end of this research, a working model of CAVs under attack will be offered, showing their resilience to safety-compromising attacks and the effect of employing different defences.

1 Introduction

Cybersecurity is a key area of concern due to the increasing levels of connectivity between different devices as well as the internet, with CAVs being one of these cases. In the case of cyber-physical systems, where the computer is controlling a physical system, this is more critical as failures here can lead to injuries and even loss of life, so the safety and security of CAVs will be the focus of the research.

1.1 Connected Places Catapult

My industrial funding partner, also known as CPC, has stated that their aim is to provide innovation as a service, connecting different businesses together and bringing them together under the umbrella of their company. CPC is a non-profit company which looks at a range of sectors, including communications, net-zero, smart cities and most relevant to my project, intelligent transport systems (CPC, 2019).

The main point of communication with CPC is Dr. Alan Peters, who is usually the most present at my meetings with CPC, aside from others such as Thomas Levermore, who are engineers involved in the CAV projects at CPC.

CPC is interested in the project from the smart cities angle and have an interest in seeing the results of my research, particularly in the form of a model or tool that will allow evaluation of cyber-resilience in CAVs.

The role of CPC in this project is to work alongside City University, as the funding was obtained jointly when EIT digital was still a partner in the project, until they withdrew in 2022 due to post-Brexit contract negotiation issues. CPC will attempt to give me help or advice where possible, as well as looking at my current progress. Their expertise in this subject is helpful, especially their connections with automotive companies, and have offered to set me up with their more experienced engineers if more technical advice on my subject is required.

1.2 Connected & Autonomous Vehicles

As connectivity increases for devices, their communications become more important, as these now present a larger attack surface for vulnerabilities. In CAVs, these communications include:

- **Vehicle-to-Vehicle (V2V):** Communications between vehicles, allowing the exchange of data to get more information on surroundings as well as being able to inform other vehicles about decisions being made that can affect traffic. These communications are short range and will have low latency.
- **Vehicle-to-Infrastructure (V2I):** Infrastructure in this case includes Roadside Units (RSUs), which communicate with vehicles to get information regarding the status of vehicles, traffic and more. The RSUs can then communicate with one another, so information on things such as traffic can be sent across larger areas, making more vehicles aware of upcoming situations earlier so that safer, traffic-easing decisions can be made early on.
- **Vehicle-to-Everything (V2X):** Vehicles will ideally be able to communicate with everything, having long and short-range communications. This would open a range of communications for CAVs, including Wi-Fi, cellular networks, vehicles and more.

This can bring in devices such as the GPS (Global Positioning System) receiver and various other sensors.

These areas of communication improve the amount of information that can be exchanged with the vehicle, including warning messages (i.e., for traffic) and those that notify other vehicles of the actions of others. The components such as Wi-Fi will also give internet connectivity and allow vehicle users ability to access networks and use a wider range of services in the vehicle, especially now that the CAV does not require a driver, entertainment can become more of a focus.

The judgement of drivers is limited by factors such as their reaction time, as there is only so much precision a human being can have, and with fast reactions required, they simply cannot think fast enough to react correctly in such a short amount of time. However, computers are known for being significantly faster than human beings and can be programmed with the necessary information and actions to perform accurate tasks.

From this, the idea of having a computer replace the human behind the wheel comes about, and the idea of automating the vehicle becomes interesting, as a computer can now act quicker and process more information at higher rates with greater accuracy than a human. From this, various companies have gone on to develop autonomous vehicles, which can include features such as automatic braking, keeping distances and controlling speed.

However, the autonomous vehicle on its own uses sensors and what it can see from its surroundings (i.e., camera images, GPS, and sensor data). While my model is primarily an autonomous vehicle, the addition of connectivity will increase the attack vectors and present higher risks.

By combining the autonomous vehicle and the connected vehicle, this results in CAVs. These do not yet exist officially, with all of them being in testing stages and having distinct levels of automation depending on the car manufacturer. Therefore, in my research, I had to look at CAV not only as a whole, but to look at the connected and autonomous sections separately, as these already exist.

Level of Autonomy	Description
0	There is no automation at this stage, so the driver performs all the tasks, with no assistance, and therefore can be held responsible for their part in any incidents that may occur because of their judgement, such as accidents.
1	The vehicle is not autonomous but can assist the driver in their task, so that certain manoeuvres can be performed smoothly. This may only be in a certain direction (i.e., lateral, or longitudinal) and the driver can immediately take full control whenever necessary.
2	Here, there is partial automation, where the vehicle performs part of the driving task as opposed to simply assisting the driver. The driver can then perform the rest of the task and is again capable of taking back control whenever necessary.
3	From this level onwards, automated driving systems are incorporated, which can perform an entire driving task or manoeuvre while active. It can now issue requests for a user to intervene if it feels the driving conditions are outside of its design capability, with the user taking control back whenever they wish.
4	The second highest level of driving automation, where the vehicle systems can perform entire driving tasks and can function within certain environments. The vehicle can delay giving control back to the user to ensure safety and ensures minimal risk. The driver will still rarely need to take control in situations where the vehicle cannot perform tasks.
5	Full automation is the end goal of the self-driving car. This means that the vehicle can effectively perceive the environment under all circumstances and require no driver input whatsoever no matter where it is. The driver can request the system to disengage so that they can drive when it is safe, but this would usually not be necessary.

Table 1: CAV – Levels of Autonomy (SAE, 2021)

Looking at Table 1, I can see the different levels mentioned before, as not all manufacturers are working at the same level, as given by the Society of Automotive Engineers (SAE). Up

until level 4 autonomy, the driver is still able to take back control, as a failure in the functionality of the CAV would be lethal if the driver were unable to act at all, which could become a reality in level 5 autonomy.

The autonomous section of the vehicle means that it is heavily reliant on artificial intelligence, more specifically machine learning. This helps to classify information such as images and identify what the vehicle is seeing, which is important for the motion planning of the vehicle. This system has several weaknesses, including its vulnerability to a class of attacks known as adversarial examples (Choi, 2021). These attacks alter the images in such a way that causes the machine learning model to incorrectly classify the objects on the image, so the vehicle makes an incorrect decision due to misidentification of the objects in question, meaning that a more resilient system is needed for this.

Communication	Description
Vehicle-to-Vehicle (V2V)	This is the communication between vehicles, exchanging information with one another such as their location, information gained from other vehicles and their status. This is usually communicated short range via dedicated short DSRC or cellular communications.
Vehicle-to-Infrastructure (V2I)	A vehicle must know information about infrastructure as well, which can include RSUs which have gathered information about different vehicles and the traffic. The infrastructure can also include the relevant signs and traffic lights that aim to ensure safe and efficient driving. These help the vehicle know in advance about changes in signs as well as incidents further up ahead for timely decision making.
Vehicle-to-Everything (V2X)	Communication of vehicles with everything in the surroundings. A special case of this is with everything on the cellular network, known as C-V2X, using mobile networks for a special form of vehicular communication as part of a new 3GPP project.

Table 2: Vehicular Communications (El-Rewini *et al.*, 2020)

As seen in Table 2, another area of importance is that of the communications, offered by the connected vehicle, which allow for exchange of information between different areas such as vehicles, infrastructure and various other entities or devices. This also means that another device, even a malicious one can exchange information with a vehicle, gaining access to sensitive data or even sending invalid data to the vehicle to cause it to make incorrect decisions. In the worst case, these communication channels could even be jammed shut to prevent any legitimate connection or communication, leaving the vehicle with little to no data for decision making (El-Rewini *et al.*, 2020).

The different architectures inside the vehicle are important, as this is where the subsystems controlling different vehicle functions are, also known as ECUs (Electronic Control Units). If an attacker successfully navigates their way into the in-vehicle network, they may be able to take control of one or more ECUs, sending messages that can change the behaviour of the vehicle on the road. Different architectures including the regular gateway-based architecture, domain-based and zonal architectures, each of which has their own advantages and disadvantages (Dibaei *et al.*, 2019).

If I am to look at these architectures, the focus will be on their resilience, meaning that I need to see which architectures are more effective overall, especially when it comes to preventing the progress of an attacker through the in-vehicle system, which would mean I could determine the resilience of different architectures.

In CAVs, the areas of concern when looking at security can be seen as the following (Ross, McEvelley and Carrier Oren, 2016):

Confidentiality: Anything which involves knowledge of various system functions, processes, or data (i.e., user information and location). Rules should be in place to govern what level of

confidentiality any user has access to, but confidentiality will not be the focus or primary concern in this project.

Integrity: Anything involving creating, modifying, or destroying elements or information in a system, which includes but is not limited to data. Integrity policies will control how or what can be manipulated in a system, which is particularly important in CAV where manipulating elements can change the behaviour of the vehicle itself.

Availability: The continued function, readiness, and accessibility of different services/elements of the system, which includes but is not limited to communications (i.e., ability to communicate with the CAV). This is particularly important as a CAV being unavailable could make it unreachable or even result in it completely breaking down on the road.

Integrity and availability are the ones that are of major concern for any cyber-physical system including CAV, as illegitimate messages or a lack of available resources could cause a vehicle to make poor, incorrect, and even dangerous decisions. These attacks would affect vehicle safety and even put passengers in danger, meaning that I need to make sure that any method of cyber-resilience applied to CAVs needs to have a strong emphasis on integrity and availability.

Sensors	Explanation
Camera	Cameras are image sensors, often used in vehicles with level 2 automation and higher, giving a 360-degree view around the vehicle. They are vital for tasks involving perception, as they are used to gather information for sign recognition and lane detection. They have been considered as a means of replacing LiDAR sensors, but they do not perform well under harsh weather conditions, so they are usually better in combination with other sensors.
LiDAR	LiDAR is a light sensor which uses light-based signals to probe the environment. Measurements are made based on the signals that are reflected to the sensor, such as the time taken to return, which can be used for obstacle detection when the relevant ECUs in the vehicle process the data.
RADAR	The radar is a sensor that sends out electromagnetic waves (radio or microwave) to detect objects, using the reflected signals to measure their distance and speed. In CAVs, they have a range of uses, such as blind spot monitoring, lane keeping and distance control.
GPS	GPS uses a satellite system placed in space by the US. By receiving satellite signals the GPS system can triangulate its position on the global map, allowing a vehicle to plot a route from its current location to a desired location using map data stored in the GPS device.
TPMS	This is used to monitor tyre pressure, so that it can be known when it is time to re-inflate the tyres, as well as changing speed according to the state of the tyres to limit damage to the vehicle.

Table 3: Vehicle Sensors (Malik and Sun, 2020)

Some of the sensors in Table 3 show how the vehicle gathers information from the outside world without its communications (i.e., connectivity), meaning that interfering with these sensors can also disrupt the correct functioning of the vehicle. It is important to not only have sensors of decent quality, but to also have systems in the vehicle that are able to detect incorrect sensor data or malfunctions so that the vehicle can be kept safe. These could be modelled to look at the effectiveness of detection, as well as the significance of attacks on different sensors and the impact of attacks on other systems as a result.

In this report, I investigated the cyber-resilience of CAVs against cyber-attacks that can have serious safety implications. This means that it was necessary to model a vehicle that is being exposed to hazards which could be happening either at the same time or due to an attack.

I achieved this using a stochastic modelling software called Mobius, where I used parameters that will increase the probabilities of delays, accidents, or late hazard detections and more to observe the significance of cyberattacks on CAVs. I then looked at which vehicle areas are

most affected and develop more detailed models of these specific areas to assess their specific cyber-resilience.

1.3 Problem Statement

The problem that I aim to solve is the lack of proper modelling and evaluation of cyber-resilience with respect to vehicle safety. Due to CAV being a cyber-physical system, any effects on the computer elements can translate into real-world physical safety issues.

My work is centred upon the development of a series of stochastic models which should capture both the effects of cyber-attacks on different functional components of CAV and the implications for the physical world, e.g. by modelling the accidents of the CAV.

Stochastic models, as any model, abstract from reality and represent CAV operation in terms of events which affect the state of CAV component, the operational environment in which CAV operates. The stochastic models allow one to conduct an analysis based on events which occur with the CAV – either accidentally (e.g. due to hazards in the environment) or caused by malicious actors (adversaries) who exploit existing vulnerabilities of CAV components – and studying the chains of other events which may result from the initial events.

The more specific problems that are addressed in terms of modelling are that most existing work does not quantify CAV safety and security, whereas the aim here is to develop a working abstract model of CAV, as well as attacks and defences which can be assigned parameters. These can then be used for simulations which will return the risk of a catastrophic failure (i.e. a car accident) over a period to see how on-road safety of a vehicle under attack varies, with or without defences. By having neither attacks or defences, it is possible to have a baseline to compare the risk levels to better evaluate effectiveness of different attacks and defences.

Stochastic models rely on a set of parameters (e.g. rates of occurrence of various events), probabilities, which branch the chain of events between alternatives, etc. More on this is provided later in the thesis.

When it comes to the selection of parameters, my primary methods will be as follows:

1. Use of existing parameters, such as those from papers or datasets that I can estimate and that can be integrated into the relevant parts of my model (i.e., as probabilities, parameters of probability distributions, etc.). These existing parameters were obtained from (Popov *et al.*, 2022) as this already had an existing SAN model of a vehicle.
2. When no credible sources exist and point estimates of some of the parameters are difficult or impossible, then sensitivity analysis would be applied for those parameters, in which a range of “plausible” values will be used. The models will be solved for so derived parameter values. The judgement about plausible ranges is my own consulted with my supervisors.

The following research questions were addressed during this project and can be considered the original contributions of this work:

1. **How will CAVs be modelled to quantify safety in the presence of cyber-attacks?**
Contributions: The work offers specific and detailed models of how successful attacks on CAVs affect failure modes and accounts for uncertainty in model

parameters. The models reflect a CAV architecture including elements such as the perception and safety monitor, building on similar cyber-physical systems developed in prior works by my supervisors.

2. How useful are these quantitative models, given that they rely on numerous parameters and what are their limitations?

Contributions: I address the issue of obtaining plausible model parameters that will reflect CAV in a realistic manner and validate the models against previous models. The impact of uncertainty in different parameters is also studied.

3. How is the proposed methodology to assess CAV safety in the presence of cyber-attacks in line with realistic CAV architectures?

Contributions: The level of abstraction of the model is changed as I progress, creating one that groups the failure modes such that I have ‘partitions’ to reflect different components such as the safety monitor, giving an architecture in which components are linked and can be attacked separately or together.

4. What methodology can be used to assess the safety impact of simultaneous attacks on multiple CAV components?

Contributions: The model is designed such that components can be attacked individually and separately (i.e. enabling and disabling attack effects on components) to compare the effects, which is significant in originality as similar studies have not been conducted before, with an early version of it having been published in DESSERT’2022. Later development also allowed for non-linear behaviour to be observed when subjecting multiple components to attack.

5. Considering the research questions so far, can the newer methodologies be applied to layered models of defence such as defence-in-depth?

Contributions: The methodologies used are applied specifically to the CAV domain and are applicable to a wider range of cyber-resilient, cyber-physical systems beyond just CAV (e.g. UAVs).

6. Can a methodology be developed for quantitative modelling of intrusion tolerant CAVs?

Contributions: The development of CAV models where intrusion tolerance is applied to each of the components. This is then extended to have a more dynamic approach (i.e. proactive intrusion tolerance) as both literature and experimentation suggested that a static approach cannot account for future adversaries that may be unpredictable. The mechanisms for defeating dynamic intrusion tolerance have also been observed with a view to further publication on this work.

2 Background & Literature Review

In this section I looked at the background of CAVs and conducted a review of the relevant literature through the method of systematic literature reviewing, as specified by (Kitchenham and Brereton, 2013), by first finding a sufficient number of relevant papers for study through manual searches, selecting papers and excluding those that did not pertain directly to CAVs, AVs or their safety and security. The references of these papers were then also checked for more relevant sources, as I collected more sources on the various parts of CAV relevant to my subject. These included machine learning, the in-vehicle network, vulnerabilities, defences, and other components (i.e., vehicular safety or communications). While various sources of literature were used, the main ones consisted of IEEEExplore, arXiv and ScienceDirect. A more extensive version of the literature review can be found in Appendix 12.2, as this section will mostly go into topics that have been directly used in the work.

2.1 In-Vehicle Network

This section addresses the components of the network inside the vehicle, used to communicate data between different subsystems which are used to perform different tasks, as well as different protocols and connections used to do this.

2.1.1 Electronic Control Units

Increasingly new features are being integrated into the vehicle. These features need to be controlled properly in an electronic system, so embedded systems called ECUs were introduced in order to make the tasks easier (Alam et al., 2019). Each ECU takes sensor inputs which are processed to get data for the relevant ECU to perform its task. This is not always direct, so one ECU could have the necessary data and communicate it to another ECU where necessary. Every ECU is a small computer connected to a sensor as well as other ECUs, and an actuator through which the software decisions are converted to physical actions where needed.

2.1.2 Controller Area Network (CAN)

The CAN bus was invented in the 80s by Bosch as an inexpensive means of connecting nodes (i.e. ECUs) in the vehicle along a single shared line, where 0s are considered to be dominant (i.e. higher priority bits) while 1s are considered recessive (Sharma and Möller, 2018). This system of bits is used for bus access, using CSMA/CA (Carrier Sensitive Multiple Access with Collision Avoidance), where more zeros in the ID section of the CAN communication frame gives it higher priority, so that higher priority frames can communicate in real-time with less traffic.

This form of communication, also known as bit arbitration, is an efficient standard for in-vehicle communication, as it is possible to assign more zeros to the more important frames, such as those which involve safety-critical systems. There is no central host in the CAN bus, with different ECUs and components being able to easily communicate with one another over the CAN bus, which is currently considered the default in-vehicle bus.

The key capabilities of the CAN bus consist of the following (Forsberg and Hedberg, 2009):

- Any node/ECU should be able to transmit messages across the bus if it is idle.
- The worst-case message delays should be possible to calculate to determine the guaranteed latency.

- It must be possible to add or remove devices from the bus without changes having to be made to the hardware or software.
- A priority system must be in place so that high priority messages get access before lower priority ones.
- Any messages transmitted on the CAN bus should be received by any node connected to the CAN (i.e., broadcast to all).
- There must be error detection and retransmission capability of all messages, with distinction between the types of errors (i.e., temporary and those caused by a defective node).
- There must be arbitration to determine which node sends first by checking their priority if two nodes intend to transmit at the same time.

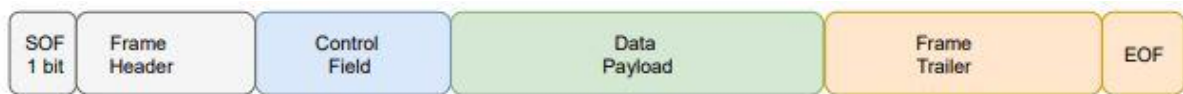


Figure 1: General CAN Frame Structure (Aliwa et al., 2020)

The general structure of CAN frames is shown in Figure 1, consisting of the frame start, header, control field, payload, trailer (i.e., CRC for error check) and the end of file. This can then be broken down further to specify the individual bits of a standard frame or extended frame.

The CAN protocols are as follows (Aliwa et al., 2020):

- High Speed CAN (CAN-H) has bitrates between 125Kbps-1Mbps, being used in real-time applications that are critical such as the engine and brakes, mostly in the powertrain and chassis domains.
- Low Speed CAN (CAN-L) has bitrates from 5Kbps-125Kbps, being used mostly in the body domain where functions are non-critical (i.e., doors and windows).
- CAN with Flexible Data rate (CAN-FD) can go up to 10Mbps, being used for critical applications that work in real time, much like CAN-H, in the powertrain and chassis domains.

The frames in the CAN bus protocols can be grouped into the following four (Aliwa et al., 2020):

- Data frames, which carry data payload, such as information from sensors.
- Remote frames, which are used to request transmission of data from another ECU. There is no data field in this frame.
- Error frames are used to indicate that there is some sort of error in the bus.
- Overload frames are used by an ECU to state that it is too busy to receive a message from another ECU, so it will have to wait.

The original CAN bus was created with no consideration that there could be outside communication and therefore any access to it by any form of attack, meaning that security was never considered. Therefore, this is an area of the vehicle that contains many vulnerabilities that will need to be evaluated, especially as there are other in-vehicle buses in existence that could offer better options, such as FlexRay.

The authors of (Aliwa *et al.*, 2020) explore the in-vehicle network, mostly the bus protocols (i.e. CAN, FlexRay and LIN), vulnerabilities and defences. The CAN bus and in some cases FlexRay are assumed to be the default in-vehicle buses for the critical parts of the in-vehicle network. It is stated that the CAN bus did not initially consider security and consists of low or high-speed CAN buses depending on how critical communications are. Modules such as the telematics unit and radio are connected to both buses which means they could be used to access critical systems in an attack. The CAN itself is vulnerable to attacks such as DoS, impersonation (i.e., fake ECUs) and false messages which should be modelled as they can have far reaching effects that could even allow control of the in-vehicle network. Proposed mitigation methods include cryptography and IDS, which could be used to detect the attacker. These are limited by the amount of overhead, cost and data required to implement them, as well as changes that may need to be made to the in-vehicle bus. The CAN should be considered the default in-vehicle bus, and variations on CAN such as CAN-FD or CAN-XL could hold the solutions such as the added security needed to improve the CAN bus security.

The in-vehicle network is clearly a critical area that will need to be part of the CAV model, even my focus. The following gaps were determined:

1. How effective are defences implemented in CAN bus and ECUs?
2. What are the safety implications of attacks on different ECUs?

Therefore, part of my modelling was done such that the failure modes became partitions like ECUs, or I had defences that modelled the effect of accessing the CAN bus.

2.2 Sensors

Sensors are one of the most vital elements to CAVs, acting as the starting point of perception for the vehicle. Without these, there would be no information on the surroundings for the vehicle to process, meaning that it would be making decisions while blind to its environment. Here I looked at the key sensors and their functions. Sensors are known to work across different ranges due to their limitations and can be described as the following (Qayyum *et al.*, 2020):

- Proximity sensors, which have 5m range, are typically ultrasonic sensors used to detect obstacles at close range, particularly when the vehicle is moving very slowly. These types of sensors can help in tasks such as parking, as they can make sure that a safe distance is kept from nearby obstacles when adjusting the vehicle position.
- Short-range sensors have 30m range, being used in forward cameras, backward cameras as well as in short range radar (SRR). Forward cameras can identify upcoming objects ahead, while backwards cameras work similarly for objects behind. Blind spot detection is also a notable application of SRR.
- Medium range sensors operate from 80-160m, consisting of LiDAR or medium range radars (MRRs) that can be used for detection and collision avoidance.
- Long-range sensors can detect obstacles up to 250m ahead, working at high speeds and collecting additional data from surrounding devices or communications to allow for cruise control, where the driver does not need to focus as much.

In this section the focus will be on LiDAR, RADAR, GPS, and cameras, as these are the best known and more common sensors for CAVs, with some mention of the additional sources of

sensor data which help to improve the overall reliability of environmental perception data gathered by the vehicle.

2.2.1 Light Detection and Ranging (LiDAR)

LiDAR is a type of sensor which uses laser light to get in-depth, 3D data points regarding different objects in the environment surround a vehicle. By creating a 3D map of the surroundings, LiDAR allows the vehicle to perceive and navigate its surroundings in a safer manner, given that it is also considered a more reliable class of sensor with accurate object detection (Cao et al., 2022). The wavelengths of LiDAR range from 10 μ m-250nm, meaning it can go from infrared to ultraviolet light (Pham and Xiong, 2020).

The pulses of light sent from the LiDAR will hit nearby objects and be reflected back to be received by the sensor, using the time taken to return in order to determine the distance between the sensor and nearby objects (Parkinson et al., 2017). LiDAR can provide 360 degree view (all-round) with over 16 channels, as well as the rotation mechanisms being classifiable as mechanical, semisolid-state and solid state (Dibaei et al., 2019). If the signals can be prevented from reaching the sensor or tampered with, this could be an area of critical concern as the ability of the vehicle to perceive the environment has now been adversely affected.

2.2.2 Radio Detection and Ranging (RADAR)

Using EM waves in the radio or microwave range, it is possible to apply the same idea as LiDAR by calculating the distance based on time taken for a reflected waveform to reach the radar sensor, which can serve various purposes such as helping with lane keeping and distance control (Pham and Xiong, 2020). It is also possible to use millimetre-wave radar, which can penetrate smoke and dust, where other wavelengths may not be able to, but the resolution and accuracy of the radar can still be weak, meaning that it should ideally be combined with other sensors to get a more accurate overall picture of the surroundings (Dibaei et al., 2019).

2.2.3 Camera

The camera is an image sensor for the vehicle and can be used for object detection, possibly as a cheaper alternative to LiDAR, but can perform badly in certain weather (i.e. rain and fog) (Pham and Xiong, 2020). Due to this, they are usually paired with other sensors, with their main purpose being identifications of things such as traffic signs, and are either mono or stereo vision, with the difference being that more cameras are needed in mono than in stereo (Parkinson et al., 2017). As the sensor is strictly image based, much like the eye, the quality can depend on the weather or visibility.

2.2.4 Global Positioning Systems (GPS)

This sensor is for navigation, being based off a series of satellites in space owned by the US government which transmit high frequency signals that can be detected by any device on Earth which has the ability to receive GPS signals, such as smartphones or vehicles that have GPS receivers built into them (Pham and Xiong, 2020). The receiver in any device is able to use the signals from the GPS satellites to compute its present location to within 1m of accuracy, but in the case of obstructions that could weaken the signal from a satellite, more satellites will make it possible to continue to get an accurate position (Parkinson et al., 2017). There is no means of verifying if the GPS signal is authentic, as the encrypted signals are for

the military only, meaning that it could be possible to compromise the integrity of the GPS receiver.

It can be seen in this section that the perception is an integral part of the vehicle, acting as the eyes an autonomous vehicle needs in the absence of a driver. Potential research gaps:

1. What are the safety implications of attacks on perception modules?

Therefore, this is a strong argument to have perception as an element in my stochastic models, as it is directly vulnerable and I need to evaluate its safety implications.

2.3 Architecture

A fundamental part of the CAV structure is its architecture, as this is how different components of the system will be joined together and communicate. It is important to have an efficient architecture with good security as well as allocation of resources such as bandwidth.

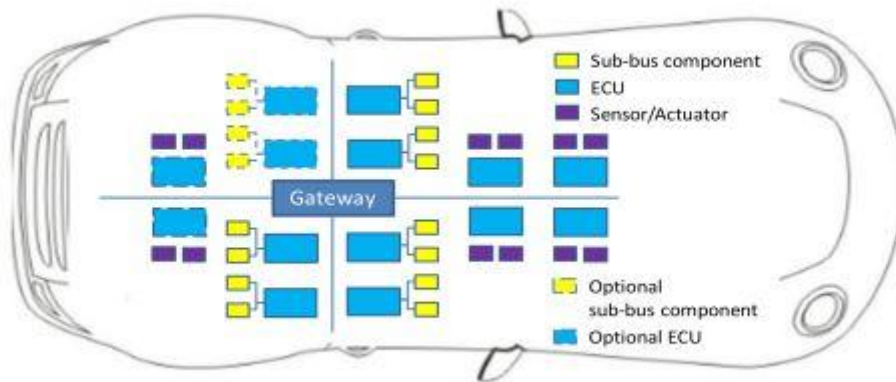


Figure 2: Gateway E/E Architecture (Dibaei et al., 2019)

The traditional in-vehicle architecture in Figure 2 is *gateway-based*, consisting of ECUs all connected to the gateway, each with the necessary sensors, actuators or other sub-bus components. However, as the default in-vehicle bus is assumed to be CAN, the CAN bandwidth needs to be shared between all ECUs (Dibaei et al., 2019), meaning that due to the lack of discrimination in bandwidth allocation, high-priority communications by key ECUs could be delayed if the bandwidth of the bus is already being used up by others, suggesting that a better architecture is needed.

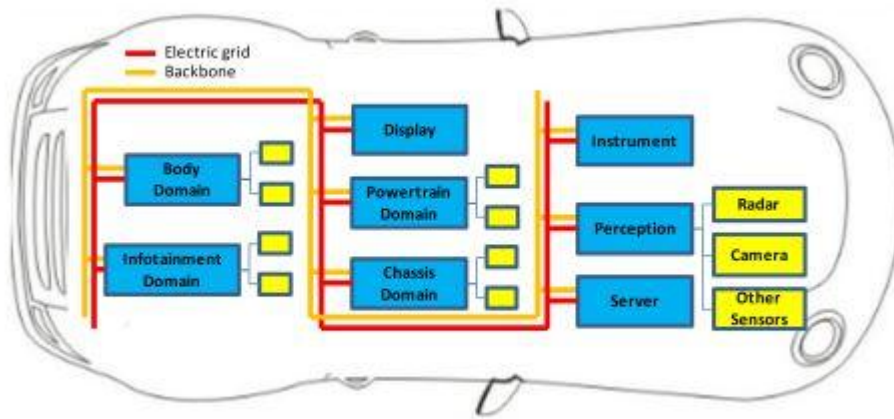


Figure 3: Domain E/E Architecture (Dibaei et al., 2019)

The proposed alternative in Figure 3 is *domain-based*, with ECUs being grouped according to similar functionality, each under a relevant domain controller, so the bandwidth is allocated to each domain in advance. Their power is linked along the electric grid of the in-vehicle network, as well as a network backbone for the allocation of bandwidth and power (Dibaei et al., 2019). Therefore, each domain now has more bandwidth and power allocated which allows for more complex vehicle functions. Additionally, the sensors can be connected to the domain controller ECU so that sharing bandwidth with the other ECUs in the domain is no longer an issue (i.e. communicating sensor data between ECUs in the same domain), and data communication inside the domain will not affect the overall bandwidth of the in-vehicle network or its backbone (Dibaei et al., 2019).

In more detailed domain-based architectures, CAN, LIN, and FlexRay are assumed to be the more popular bus protocols the vehicle may use. In this example, the following domains were specified (Aliwa et al., 2020):

- **Powertrain:** Critical functions such as the engine and transmission ECUs, which need real-time response and will therefore use a high-speed or flexible-data rate CAN (CAN-FD).
- **Chassis:** This consists of ECU such as brakes, suspension, and steering. These provide real-time and safety critical functions to the vehicle, so the CAN will once again be one that can provide high speeds/data rates, or even the FlexRay protocol instead.
- **Body:** These are not particularly critical systems, as the ECUs will deal with things like lights, the dashboard and windscreen wipers. This domain either uses low speed CAN or the LIN as its bus.
- **Telematics & Infotainment:** Mostly used to manage communications, data and entertainment in the vehicle, no bus is specified, but it has access to the gateway that links to the other domains, making it a potential security risk if compromised.

In addition to these, the different sensors such as light sensors and cameras are also linked to the vehicle through the gateway, as their data could be needed by the different domains to take the correct actions depending on their function. Therefore, any potential weaknesses and threats to the sensor system should also be considered when evaluating any security.

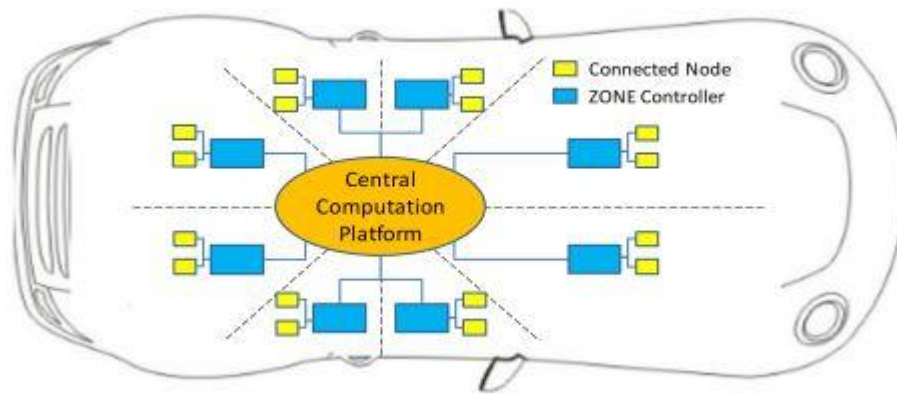


Figure 4: Centralized E/E Architecture (Dibaei et al., 2019)

Another architecture proposed in Figure 4 suggests a *central computer* instead of mini computers for each ECU or domain, so all tasks and functions exist in the computer domain, which has access to all the sensors and actuators at once (Dibaei et al., 2019). This allows for total sensor fusion (i.e., combined data from all sensors at once to visualize the environment), but the demands on data communications will increase as a result, as well as adding the possibility of a single point of failure or compromise for an attack on the in-vehicle network. The different parts of the network are divided into sub-network components according to their position in the vehicle or network properties to make them more efficient and are each controlled by a zone controller (Dibaei et al., 2019).

While these vehicular architectures have all shown to have different strengths and weaknesses in general, their security is yet to be rigorously evaluated in the form of models, which could be used to find out which architecture has the best cyber-resilience.

The research gaps identified here were as follows:

1. What are the safety and security capabilities of using each architecture?
2. Which architecture is most effective?

Looking at these, I can tell that some sort of architecture must be modelled to effectively represent the in-vehicle network, so my failure modes need to be grouped in a way that allows partitioning at sufficient level of abstraction to reflect an architecture.

2.4 Safety

In vehicles, safety is a key element, as car crashes are already a known issue that exist due to human beings, so CAVs must aim to reduce the number of crashes, therefore meaning that safety measures should be implemented to reduce the rate of accidents in autonomous driving compared to regular driving, as it boils down to the statement by (Koopman, 2022) which asks what level of safety is truly safe enough. Specifically, the components that deal with or could affect safety when not functioning properly are known as the safety critical systems, with models on security taking these systems into account and evaluating their safety with or without attacks, ideally as models (Popov, 2015).

Some AV vendors assert that their products already demonstrate better safety than the vehicles driven by human drivers. Waymo, one of the leading AV manufacturers in the USA

claim that the accidents with their robotaxis are less severe than the accidents with human drivers (Favaro *et al.*, 2023).

One of the issues with CAVs is that safety guarantees cannot easily be made, meaning that unless CAVs are able to promise and deliver on improved safety, as well as sufficient security to maintain the safety both in the event of a fault or compromised system, it is unlikely that drivers will have any interest in switching to autonomous driving (Shalev-Shwartz, Shammah and Shashua, 2018). In addition to this, any safety solutions implemented must be scalable so that if there are millions of CAVs on the road the solution can be scaled up to suit all of them without incurring massive costs or becoming exceedingly complex.

To achieve better safety, the Responsibility Sensitive Safety (RSS) module is written as a policy consisting of the parameters of a vehicle such as its speed, distance, and acceleration to ensure that the vehicle is always behaving safely. The RSS essentially works as a mathematical model to describe what a safe driver is to the CAV, working as a set of rules that it must follow, which can be broken down into the following (Shalev-Shwartz, Shammah and Shashua, 2018):

- No hitting road users from behind.
- No reckless driving (i.e., sudden cutting into lanes).
- Wait for the right of way to be given on roads, do not try to take it.
- Be extra cautious where there is limited visibility.
- Avoid causing an accident wherever possible provided it can be done without causing another one.

Obviously, these rules are implemented as a mathematical formula that makes use of the relevant subsystems such as sensors and ECUs. The parameters would need to be well chosen, as having very relaxed parameters could reduce overall safety by having weaker reactions, while a conservative policy (i.e. too much safety) could result in a very impractical and slow driving experience (Oboril and Scholl, 2021).

For a constant acceleration, the minimum distance can be modelled using the velocity of the vehicles ahead and behind the driver, as well as the maximum possible acceleration and deceleration, so that if the distance is less than this value it is considered unsafe (Oboril and Scholl, 2021). It is important to have a good deceleration as quick, hard deceleration could result in being rear-ended while slow deceleration could result in an unsafe distance or a collision with the front vehicle, meaning that optimal values should be determined.

Therefore, the safety of the vehicle is based off the safety-critical ECUs such as those controlling braking and steering, the in-vehicle bus as and the RSS that governs the safety. These features as well as any data gained from the surroundings by methods such as sensors, communications and how they are processed by the machine learning model will be important in modelling when considering safety and security together.

The authors of (Li, Wang and Zhang, 2021) propose an attack method against cyber-physical systems such as CAVs, known as Chronos. This attack aims to disrupt synchronization and add delays to the perception system, which can in turn affect planning and control as the sensor data communicated is not all synchronized, so the fusion will no longer work as effectively as usual. Attacks on the sensor fusion will interfere with the timing between the

sensors, while delays added to control will cause deviations in the results compared to what they really are over time. Attacks were conducted on various levels, against both drones and vehicles which were simulated on computers with simulators and datasets. The experiments showed that the attack caused serious deviations sometimes completely outside the intended trajectory which could result in serious accidents. Authors recommended more studies and modelling of this attack to better mitigate it in future as it can post major risks to cyber-physical systems.

It is noted that in (Lisova, Sljivo and Causevic, 2019) the authors have found 33 sources of literature and have indicated that safety and security co-analysis is a growing field in the area of engineering. Several papers were presented by the authors in their reviews, which indicated that approaches to safety and security should include models such as HAZOP, FMVEA, STRIDE or stochastic modelling (i.e., SAN models and Markov processes). The ISO 26262 standard could also be used as the base of the framework for potential analysis or a guide to developing other models in future based on its requirements. All the models are worth looking into and based on how I wish to analyse (combined or separately) should then choose the appropriate models.

Continuing from this survey of suggested models, (Schmittner, Ma and Smith, 2014) tell us that the existing technique of FMEA only looks at faults and fails to account for security, which is becoming increasingly important, especially in vehicles, so FMVEA, which considers vulnerabilities is proposed as a better choice. This looks at the failures, the vulnerabilities, and their effects. Following the principle of FMVEA, the authors focus on a specific component of the AV in this case, the telematics unit, due to its connection to both buses, allowing potential access to safety-critical systems. In the context of the telematics unit, over-the-air (OTA) updates and Bluetooth are considered to have the highest risk, having high severity and attack probability (their product is the risk), with the possibility of an attacker taking control of the vehicle. This form of risk analysis, or other models such as STRIDE (Schmittner, Ma and Smith, 2014) could be used to prioritize the level of risk to see which attacks to focus on and considering defences for. The focus on telematics also gives us a potential starting point for modelling attacks, especially remote attacks.

Focusing more on the actual AV, the authors of (Shalev-Shwartz, Shammah and Shashua, 2018) tell us that one of the biggest issues with AVs lies in the lack of safety guarantees. The accident rate must be lower than that of a human, which is stated to be 10^{-6} per hour, so the accident rate should be significantly lower than this, ideally around 10^{-9} per hour. RSS is introduced as a system that follows the idea of a duty of care, meaning to exercise care and caution when driving on the road. By choosing appropriate parameters these rules can be implemented in the CAV, in a way that is sound, useful, and verifiable.

RSS rules use consists of the following ‘common sense’ rules for vehicles (Shalev-Shwartz, Shammah and Shashua, 2018):

1. No hitting from behind
2. No reckless overtaking
3. Right of way must be given, not taken.
4. Be extra careful in limited visibility.
5. Avoid accidents wherever possible if it can be done without causing another.

If the model follows these rules, I could even create my own where needed provided it follows these rules well in an effective manner with plausible parameters. The white-box nature of the RSS model means its inner workings are visible (Shalev-Shwartz, Shammah and Shashua, 2018), which could potentially be exploited with the right kind of attacks which may even manipulate the information being received to check if the rule is being applied correctly.

However, it was argued by (Oboril and Scholl, 2021) that the RSS model in its current state may lead to sub-optimal results in extreme cases and suggest their own, proactive extension of RSS known as RSS+. AVs are simulated using a vehicle simulator known CARLA (Car Learning to Act) when implementing RSS+, as RSS sensors are integrated into the AV in this case, so that RSS and RSS+ could be compared. If comparing potential safety monitors and the vehicle when on the road, CARLA could be a very good simulator for us, as I could load my own maps, libraries and scenarios (i.e. hazards). The simulations show that when comparing RSS and RSS+, the distance is better kept over a period of time for RSS+ than it is for RSS in scenarios such as obstacles, pedestrians crossing and vehicles overtaking (Oboril and Scholl, 2021). Overall, the collision speeds are lower for RSS+ than they are for RSS which indicate better safety monitoring. From this I concluded that RSS+ is better at proactively dealing with safety on the road, and I could even attempt to model it as a means of mitigation for safety-related issues, though tying it in with security challenges will present its own range of issues.

(Popov *et al.*, 2022) states that AVs need to have fundamental safety issues resolved, particularly the overall safety, going on to model a vehicle experiencing hazards on the road. It is stated that 10^{-5} is the highest the accident rate should be, if considering human drivers, for AVs to be considered safe in comparison. It is impractical to try and derive arguments for safety from mileage due to time and software updates. A better idea is to model the vehicle and its components, using simulation to see the overall safety. The model can be used to see how the failure rate (i.e., risk of accidents) is for a vehicle with or without hazards (i.e., traffic and harsh weather). All hazards are assumed to have a duration, meaning they do not go on forever and the hazards occur at a fixed rate; they are either correctly perceived or overlooked in this case. Parameterization was done through sensitivity analysis and the time of each simulation being 100-9100 hours.

Results from the cases tested are as follows (Popov *et al.*, 2022):

1. Overlooked hazards are shown to have significant effect on the risk of an accident.
2. The duration of such hazards has negligible effect.
3. The risk of accidents following an overlooked hazard that is later correctly perceived are only slightly more, with 3-3.5% risk of an accident.
4. Where the safety monitoring is considered imperfect, effects are significant, with an accident being imminent with 9100 hours, as 40-50% accident probability has been reached by 2000-3000 hours, so the safety monitors must be reliable, with rates of 10^{-6} at most.
5. False alarms (i.e., detecting a hazard where there is none) are also negligible, remaining under 3% for false hazard probabilities between 0.1-1%.
6. Combining imperfect safety and falsely perceived hazards has little different from that of the imperfect safety monitor alone, suggesting that a falsely perceived hazard is of little importance either way.

It can be concluded that while some parameters only have minor effects, those related to perception and planning have the most significant impact, namely overlooking hazards or imperfect safety monitors, which mean these are where designing and choosing parameters will be most important.

Despite quantifying the level of safety, (Popov *et al.*, 2022) did not address the security component, which is very significant in modern vehicles. A stochastic model for both safety and security was created by (Popov, 2015) using Mobius for an ASIL-D security requirement device known as the e-Motor, with fault tolerant mechanisms, having two channels that detect their own failures, with their own safe and unsafe states. Monte Carlo simulations were used in a range of 1000-12000 hours, with parameters for the repair rates (safe and unsafe states) of the device being obtained through sensitivity analysis, though the analysis itself was not presented. For the simulations, as the repair rates were increased, the failure rates decreased, with unsafe failures happening later and later for a total of 200,000 simulated hours per experiment (Popov, 2015). It is suggested by the authors that probabilistic modelling is the best way to quantify the effects of cyberattacks, with probabilistic modelling being very suitable. The comparison of IDS and individually protecting devices in AVs seems a promising area to explore, which I could do in the future.

I note from (Bloomfield *et al.*, 2021) that assessments of safety for autonomous systems such as AVs, the system should adapt as needed according to what is required of it, with good analysis and testing to give confidence in its safety. The models, stochastic or otherwise should allow us to test the safety in combination with security so that I can determine the confidence in CAV safety and security. Safety modelling in (Bloomfield *et al.*, 2021) was done for a sensor with an ML component and a safety monitor added to it, mostly as templates of the requirements, as well as the strengths, weaknesses and environment. Therefore, when modelling vehicle safety and security, considering ML is especially important, due to being the key intelligent component of the system (i.e., provides the autonomous capabilities). As well as this, the adaptation and changes of the system should be considered, as different changes and future requirements could be of importance due to introduction of vulnerabilities and faults in the future.

A guide to HAZOP analysis, mainly designed for various industrial processes and systems, so that hazards when operating can be seen is offered by (Crawley and Tyler, 2015). System parameters are combined with guidewords and checklists to make sure all relevant questions are asked (i.e., consistency, control of the main parameters). Guidewords are simple phrases that have a generic meaning to describe something in relation to the system design. It must be considered that guidewords are not necessarily an effective way of rating risk, which would have to be assigned to the guidewords or done alongside them somehow for it to be properly quantified. As the CAV is a system with various functions and processes, it may be useful to look at risks and answer all important questions, especially those involving safety using such a method.

The authors of (Schmittner *et al.*, 2015) look at safety and security analysis, comparing FMVEA and CHASSIS when being applied to the automotive domain. Due to how computers and communications are becoming more intertwined, the existence of cyber-physical systems and overall systems dependability becomes more important. Major challenges in cyber-physical system development include (Schmittner *et al.*, 2015):

1. Requirements of engineering scaling up with complexity of the systems, as existing methods do not do so.
2. Categories of dependability can no longer be viewed independently.
3. Mass deployment could present issues, especially as dependencies could lead to misuse and malicious interactions (i.e., attacks).
4. Maintenance will be needed to manage new vulnerabilities, defects as well as components becoming worn out. Some remote maintenance will be needed so the risks will have to be considered for this too.
5. Context is important for such systems so it must be monitored.
6. The infrastructure of the systems should be flexible so that it can adapt as well as being easily modified in future.

Authors present a case study of Over-the-Air updates in a vehicle, which consists of the vehicle, remote user, and communication network. The security is looked at in relation to CAN and FlexRay inside the vehicle, which are found to be insufficient in terms of security. The communications of the vehicle are handled by cellular, Wi-Fi or Bluetooth usually. Authors here have excluded non-critical buses such as MOST or LIN in their analysis. Scenarios looked at include tampering with updates, false updates as well as those designed to interfere with the telematics unit. Telematics attacks are designed to impact availability in this case, either by flooding it with large amounts of firmware to crash the communications of the unit or sending false updates to block legitimate ones from going through (Schmittner *et al.*, 2015).

It was found that CHASSIS is good for the initial stages where elevated levels of abstraction and limited system knowledge is available, but FMVEA becomes suitable once there is information regarding system architecture. CHASSIS requires expert knowledge which can lead to results varying as new groups conduct the analysis while FMVEA produces results that are more reusable and comparable due to lack of extreme variations each time (i.e., due to diverse groups having different expert knowledge). FMVEA can be considered better for risk ratings as it tends to be connected to the system model when describing different scenarios (Schmittner *et al.*, 2015). FMVEA is also considered more formal, which means that CHASSIS is also more flexible even if not as consistent, so dynamic systems where threats change, or things are unclear could be better analysed by CHASSIS. It can be concluded that they both have their own weaknesses, with CHASSIS requiring a lot of judgement from experts, while FMVEA still needs to unify failure and threats. For better FMVEA, more threat catalogues are needed as this is how it identifies modes.

In (Kosari, Popov and Roy, 2022), written by me and my supervisors looked at the safety implications of individual parameters when compromising a vehicle where hazards are occurring at a fixed rate. It was found that individually compromising components is insufficient, so more than one being compromised is the way to go forward in any future modelling or experiments of this sort with Mobius.

A more recent incident was that of Tesla with their full self-driving (FSD) feature going wrong in a tunnel, when the vehicle changed lanes and abruptly stopped as if having identified a hazard such as an obstacle, which then induced a hazard in the form of a rear-end collision which then lead to a pile-up of 8 vehicles which then also resulted in traffic coming to a dead stop on its way to the tunnel for over an hour (Kay, 2023).

Safety is clearly a key element, and the description of the safety monitor in literature makes it clear this is a critical area that can be modelled as its own component that operates based on information obtained from other components such as sensors. The research gap here is simple:

1. What are the safety implications of attacks against the safety monitor?
2. How do I best protect and strengthen the safety monitor?

2.5 Vehicular Communications

This section is more about the communications of the vehicle with the outside world, with the important nodes and communication protocols needed to do this, as an increased amount of information from the surroundings, such as that from vehicles will improve the overall accuracy of the CAV when driving. Vehicles have an on-board unit (OBU) which allows for their communication with other vehicles and more (El-Rewini et al., 2020).

2.5.1 Vehicle-to-Vehicle (V2V)

This is a short-range form of communication between vehicles, exchanging information such as speed and trajectory, which could be used by safety-critical applications, so that vehicles can communicate and coordinate with one another to make more accurate, safer decisions that may even help improve overall traffic (Dibaei et al., 2019; Fowler, 2020).

2.5.2 Vehicle-to-Infrastructure (V2I)

Vehicles should also be able to communicate with infrastructure, which includes information on traffic obtained from roadside units (RSUs). Information obtained from RSUs and exchanged with vehicles helps to understand the upcoming traffic situation, such as congestion as well as changes in different traffic signs or signals (Fowler, 2020).

2.5.3 Vehicle-to-Everything (V2X)

V2X encompasses all communications, including vehicles, infrastructure, pedestrians (Vehicle-to-Pedestrian), remote servers (Vehicle-to-Cloud) and more (Fowler, 2020). Cellular communications are also part of this, being linked to the telematics unit as well as being a specific component of V2X known as C-V2X, meaning it can make use of nearby mobile base stations for communications (Fowler, 2020).

The research gaps here are:

1. What are the safety implications of attacks against different communication channels?
2. How do I protect the communications channels?

The communications channels are clearly a vulnerable point as the information that is sent to the vehicle through them can impact its decisions. While modelling communications channels and nodes themselves would be useful, the extent to which it is unknown as well as needing to essentially model the whole environment to do this effectively means that I will assume my attacks can be conducted through these channels to attack the vehicle rather than focusing on the communication attacks themselves, as I wish to directly observe the interplay between safety and security.

2.6 Vulnerabilities

Here, I looked at the vulnerabilities that could be exploited by an adversary to hamper the performance of the vehicle, alter its behaviour, and even take control of it, with the focus being those that could have safety implications.

2.6.1 Attack Surface

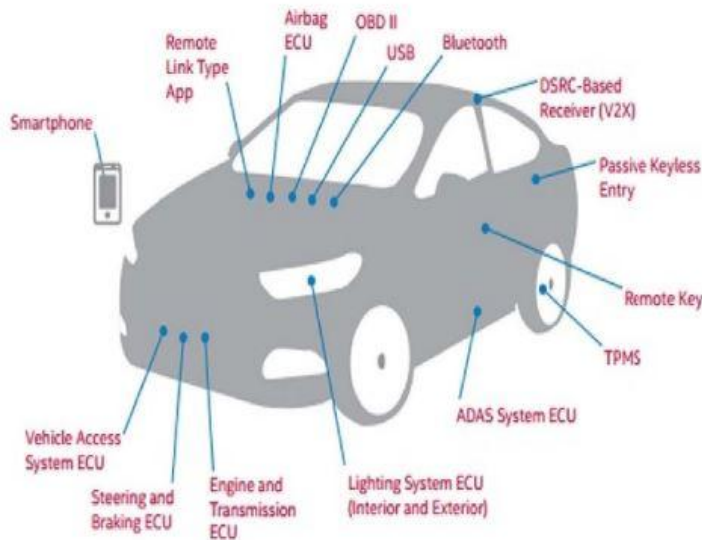


Figure 5: Vehicle Attack Surface (Malik and Sun, 2020)

The attack surface is how the attacker aims to penetrate the systems of the vehicle and gain access to further resources inside the vehicle. This is what is exposed to the outside, where the attack can either be used to cut off access to the outside or to manipulate the vehicle and its systems. Looking at Figure 5, the attack surface can be described as follows (Malik and Sun, 2020):

1. Remote connections, such as V2X communications, phones, and Bluetooth devices.
2. Ports such as OBD-II and USB.
3. Keyless entry and remotely linked applications (i.e., vehicle apps).
4. Manipulation of ECUs such as steering, transmission, airbags, and lighting.
5. Various sensors or monitors such as those monitoring tyre pressure.

In (Gao *et al.*, 2022), the attack surface and communications are identified as follows:

- **V2X:** Updates, Bluetooth, DSRC, Phones etc.
- **In-Vehicle:** USB, OBD-II, CD, TPMS and any ECU that is accessed once in the in-vehicle network.
- **Sensors:** Camera, LiDAR, GPS, etc.

The GNSS is what provides the GPS data for the localization stage of AVs. The next stage is the perception which uses the sensors, followed by V2X where all the information needed for route planning is reached followed finally by the actuators that act as the actual controls.

While any of the elements from in-vehicle can be used to invade CAN, it is also possible to perform a TPMS invasion which involves injecting attack codes into the TPMS which will activate at a certain pressure level, which can act as a sort of sleeper agent.

The proposed security methods for CAN in existence involve either encryption or authentication measures, intrusion detection (i.e. machine learning). In 2012, CAN FD was introduced as a potential future measure by Bosch, as it offers safe data transfer and error detection. However, it does not guarantee confidentiality and integrity, so attacks against these elements in the CAN are still possible in CAN FD.

Examples in the real world of safety and security failures are as follows:

- Sensor security failure in May 2016 causing a Tesla on autopilot to crash into a truck. A similar issue occurred again in June 2020.
- A failure in March 2018 of the operation system security meant that despite detecting a cyclist early on, an unmanned Uber vehicle crashed into the cyclist as the software failure to correctly predict trajectory.
- V2X communications for a Tesla were compromised in November 2016 when researchers successfully hacked a phone, obtaining the login of the user's Tesla app login, giving them remote access to the vehicle.
- In January 2018, the datacentre of GoGet was hacked, stealing a large amount of vehicle user info from the carsharing company.

2.6.2 Attacks

There are various attacks that could affect the functioning of a CAV, with the safety and security issues being primarily due to attacks that compromise the integrity and availability properties of the system. Attacks can be physical, requiring direct access to the vehicle or remote, meaning they can be done across exceptionally long distances. The general form of attacks consists of the following (Dibaei et al., 2019):

1. **DoS/DDoS:** An attacker can flood a node with copious amounts of information to use up its resources so that legitimate nodes cannot communicate with it. In a DoS attack, this typically uses one IP address whereas in DDoS multiple IP addresses can do this at once, being much more effective much more quickly.
2. **Black-Hole:** Packets are dropped so that they do not reach their destination, which means that an attacker can block off parts of network communication, which can be even more dangerous when packets are blocked at a critical point in the network, especially if these communications are intended for safety-critical systems.
3. **Man-in-the-Middle (MitM):** This attack is where a node is between two communicating nodes, without them being aware of this, allowing it to intercept, drop and modify packets at will.
4. **Replay:** A continuation of MitM, intercepted packets can be forwarded to another node at a later point, even being used by an attacker to authenticate themselves as another node, allowing it to be used for impersonation.
5. **Sybil:** This attack involves creating multiple identities that can then allow the attacker to have more control of network resources as well as greater influence which could allow for manipulation of other nodes in the network by sending them information to change their behaviour (i.e., with multiple false CAVs causing another vehicle to change its direction because of the information).

6. **Impersonation:** Quite straightforward, in which the adversary pretends to be another, legitimate node and is now concealed in the network, having the same level of access and authenticity that the legitimate node enjoys.
7. **Malware:** A malicious node could install software containing a range of malicious tools such as Trojans or viruses into the systems, which can alter the functionality of the system, and even propagate from the vehicle to other nodes using the communications.
8. **Falsified Information:** Information that is not correct is spread through the network, usually through communication systems, and could be propagated throughout a network if not detected, affecting the decisions of nodes involved, which can be very serious if it affects the driving decisions of CAVs on the road in real time.
9. **Timing:** An adversary using a rogue node can inject messages with time slots to increase delay, which can have serious implications for time-sensitive applications as they will now receive information later, resulting in later reaction times which could increase the risk of accidents.

Vehicle Component(s)	Vulnerabilities	Effects
LiDAR	Jamming using laser of same wavelength. Spoofing by sending signals back from a transmitter to deceive the sensor.	Jamming will render the sensor useless, blinding whatever parts of the vehicle rely on the LiDAR, while spoofing will cause incorrect perception of the surroundings.
RADAR	Jamming of radio waves with a transmitter. Spoofing of radio signals, such as those obtained from another radar.	Like LiDAR, the vehicle can be blinded or fooled which could lead to incorrect decisions and poor hazard perception, potentially even leading to accidents.
Camera	Blinding of the camera using bright laser light. Poor perception in harsh weather conditions. Creation of false or distorted images.	Poor weather means the perception receives poor images, which could be incorrectly identified, like if there are false/distorted images being received. Blinding renders the sensor useless and, in some cases, could damage it.
Machine Learning (ML)	Perturbed images, known as adversarial examples, being sent to the ML model, which are designed to mislead perception while looking completely normal to a human eye.	Adversarial examples will manipulate the ML model classifier leading to incorrect identification of images of the surroundings, increasing the error and reducing overall confidence (Kumar and Mehta, 2017).
GPS	Jamming of GPS receiver using stronger signals than the satellites. Spoofing of GPS using a transmitter with a stronger signal, overpowering signals received from other satellites.	The jamming of GPS will make a CAV completely unaware of its location, so the effect depends on how it is programmed to deal with a loss of GPS signal. Spoofing can result in incorrect changes in direction and manipulation of where the vehicle is going. Effects would again depend on the policy implemented for this as well as what the spoofed signals are saying, as it could do anything from stopping to even turning when it is illegal to do so.
CAN	Lack of authentication allows the injection of false messages as well as ECU impersonation. The priority scheme of CAN frames means that the bus is unusable by other ECUs while high priority frames are on it.	False ECUs or messages can be used to send instructions in the CAN bus to change vehicle behaviour, including commands like braking or steering that could be dangerous depending on the scenario. Flooding of high priority frames onto the CAN bus acts like a DoS attack meaning no other ECUs can communicate, which means the in-vehicle network is no longer operational so the vehicle cannot communicate or react properly.
LIN	Spoofed messages used to send commands alter the state of slave nodes to sleep or change their synchronization. Response collision attacks can be used to block transmission from legitimate nodes. Header collision attacks can be used to inject messages to tamper with the response of the LIN bus.	By putting slave nodes to sleep it is possible to turn off the nodes and therefore the communications. The safety implications are unlikely to be severe as LIN is not usually used for safety-critical systems (Aliwa et al., 2020) but if it were, the entire safety-critical communication would essentially turn off. Blocking transmission from nodes can lead to information not being communicated, such as the windows not opening or closing. By injecting messages, it is possible to control opening and closing of features such as the windows, A/C, or similar non-critical elements.

FlexRay	Attacks focus on the static segment, which can include: Impersonation/Masquerading Replay attacks	These attacks allow information to be sent to ECUs on the bus that could disrupt the safe and correct functions, as well as potentially allowing an attacker control of the vehicle.
MOST	Disrupting synchronization of nodes in the MOST network. Jamming of the MOST bus by constantly requesting data channels using the MOST control channel(s).	Nodes not being synchronised mean they do not operate on the same clock so they could do things at the wrong time, even reacting too early or late. By blocking the entire bus, it is unavailable, blocking all in-vehicle communications.
Ethernet	Manipulating network traffic and attempting to gain control of various nodes and switches. Overwhelming the network through physical or network-based attacks.	Control over switches can extend to control of the vehicle, disrupting the Ethernet bus and its components. Overwhelming the network is a form of DoS, which when done physically could aim to flood the circuits electrically to damage them, or network-based, the more common one which will make the Ethernet bus unavailable and therefore disable in-vehicle communications so ECUs cannot do their tasks safely and correctly.
OBD-II	By connecting a dongle it is possible to access the in-vehicle network, even updating the firmware and having access to the CAN bus (Parkinson et al., 2017).	In-vehicle access of this sort means the CAN bus can be manipulated, and it is also possible to update the firmware in a way that is malicious, either infecting the vehicle or causing it to work incorrectly. Message injection could also be possible through this port.
USB	Connecting various devices that could be compromised to the vehicle is possible through this port, so installing viruses and backdoors through the entertainment is possibly this way (El-Rewini et al., 2020).	By compromising a vehicle in this way, incorrect functions could lead to accidents because of components being faulty due to an infect network, as well as potentially propagating malware as well as the fact that the presence of a backdoor could allow for attacks that gain entry and even control of the CAV in future.
Critical ECUs (i.e., Transmission, Steering, Airbags)	An attacker being able to inject information or commands to these ECUs to control them.	Control of these could lead to incorrect function, such as in the case of steering or transmission where incorrect motion could lead to a crash. Airbag failure could be a threat to life if a crash were to occur.
Communication systems (i.e., V2V, V2I, V2X)	The communications could be flooded with data or even have false information passed through them (Hasbullah, Soomro and Manan, 2010; El-Rewini et al., 2020).	Jamming of communications could disrupt vehicle reactions due to lack of information so the CAV is less able to behave safely, not receiving the necessary messages about traffic or accidents. False information could propagate and mislead many vehicles which could cause accidents. It is even possible to send information in the form of rogue updates such as malware that disrupt vehicle functionality.

Table 4: Vehicular Vulnerabilities (Dibaei et al., 2019; El-Rewini et al., 2020)

Attacks are not always direct, as the adversary usually goes through the attack surface, and depends on what they are aiming to compromise. In order to do so they could perform a range of attacks that involve compromising different areas, but will either disrupt sensors, machine learning or aim to go for the in-vehicle network, which can be done through the OBD-II port, USB, communications or the telematics unit, a specific ECU connected to the entertainment and the CAN (Foster et al., 2015; Pham and Xiong, 2020). Looking at the vulnerabilities in Table 4, there is clearly a need for more work as well as the implementation of defences, especially as there are still some knowledge gaps when it comes to security, including the following (Parkinson et al., 2017; El-Rewini et al., 2020):

1. Vulnerabilities in telematics systems.
2. Implications of exploiting navigation systems (i.e., compromising GPS).
3. Analysis of the vulnerability and effects of compromising individual ECUs.
4. Defences for LIN, FlexRay and MOST, as well as defences to prevent attacks from different buses reaching each other (i.e., LIN vulnerability being used to get to FlexRay).

A specific case of attack through a unit attached to the in-vehicle network covered by (Foster et al., 2015) was the telematics unit, which is considered an aftermarket device (i.e. added to

the vehicle later by a separate manufacturer). The authors examine the telematics unit, particularly a unit connected to the OBD-II port, showing that it can be discovered and attacked remotely. Like most ECUs, the telematics control unit (TCU) can communicate with the CAN bus meaning if compromised it can access the in-vehicle network. If physically accessing the TCU, it is possible to take it apart, even download information it contains to access the vehicle systems in future using any credentials or passwords stored on it. Remote attacks focus on the cellular network and IP, as both can be used to send and receive requests from the TCU. These included internet-based attacks in which previously identified details could be used to log into the unit, or to use SMS to initiate rogue updates or installations that could be used to install a backdoor on the unit to access it at any time, even erasing all traces of the update afterwards. Devices could be found provided that the IP or phone number of the TCU is known; this can be done using search engines to find the ranges of IP addresses for possible TCUs, as well as the fact that certain area codes for phone numbers are reserved for TCUs in some cases which can allow for random testing and dialling to find them. By compromising the system, it is then possible to send rogue CAN packets into the in-vehicle CAN bus, which the author found that functions such as brakes and windscreen wipers could be controlled. Solutions to the TCU vulnerability included authentication of updates, better SMS security (i.e., use of blacklisting or whitelisting to control which numbers are allowed to communicate with the TCU), and even disabling administration via the network which can prevent any form of internet-based attack.

Here I saw a fair amount of real-life cases of attacks and their implications, indicating that these are the kinds of attacks I need to be concerned about and make sure my model can cover. For this I either need a range of attack models or a generic attack model where the parameters can be modified on demand to reflect all these different attacks and the way they interact with the vehicle attack surface.

For this, a generic model of attack was used which could affect different components such as perception individually or in combination with others such as the safety monitor. The attack model uses a rate parameter for attack intensity and a likelihood of success for each attack that occurs. The successful attack on any component then causes the reliability of the components in question to deteriorate. Parameters can be assigned different values such that by using a range of values we can choose plausible parameters for the attack model using sensitivity analysis, showing how changing the parameter values affects CAV safety.

For all attacks listed in this section, they can and have been modelled at the chosen level of abstraction provided using a generic attack model. Slight alterations were made to the CAV model in some cases (e.g. a layered attack that must compromise the CAN bus before attacking ECUs) to better reflect certain attacks, while the rest automatically fit with the level of abstraction which does not require too many specifics aside from suitable parameters.

2.7 Defences

This section looks at some existing defences that are either being fully implemented or are in progress, as well as their benefits and drawbacks. The limitations will be looked at as well as where they could be implemented in CAVs. These include cryptographic solutions, which consists of encryption and authentication, as well as methods to detect and prevent attacks.

2.7.1 Authentication

With lack of authentication comes the ability to impersonate nodes and send messages without any way of checking where they come from or their legitimacy in general. This is a particular issue in the in-vehicle network with the ECUs and the bus overall.

This also means that firmware updates could potentially be loaded into different subsystems without any approval needed, so cryptographic methods should be implemented in which ECUs or other nodes must authenticate themselves with a central authority that manages security using asymmetric encryption, after which communications are then set up via stream authorization which uses symmetric encryption for messages (El-Rewini et al., 2020).

Different authentication protocols have been presented in the past, including the use of hardware-based authentication as well as Message Authentication Code (MAC) being added to frames in the CAN bus in order to authenticate messages sent by ECUs in the CAN in real-time (Pham and Xiong, 2020). By using methods to securely verify the credentials of any sender on the network, attacks such as impersonation and false identities can be prevented (Chowdhury, Islam and Khan, 2019).

(Shangguan *et al.*, 2022) proposes dynamic watermarking, where controllers, such as those in ECUs add random private signals that are treated as a sort of background noise. Sensors can report these measurements and any changes in these background values could be used to detect a potential attack such as with false messages where the watermarking has not been done. The experiment done by the authors aimed to target a specific node, tampering with the original data input, and sending a false measurement to a controller. This attack could be performed against various components such as Bluetooth, Wi-Fi, and telematics. Tampering with data, in this case the yaw rate for different periods of time showed that the watermark-based detection method was very effective, being detected within 20ms when the sample rate for the detector is set to 50Hz; it is implied that this can be made faster for higher rates.

2.7.2 Intrusion Detection Systems (IDS)

A means of securing the overall networks for communications, both for in-vehicle and the wider network of vehicles on the road (i.e., VANETs) is the IDS. The IDS is a system based on deep learning which monitors the network, being a trained ML model, typically a recurrent neural network as this can compare to check for unusual behaviour that could be the result of an intruder trying to access the network (Chowdhury, Islam and Khan, 2019; Dibaei et al., 2019). There are two main types of detection used in IDS (Dibaei et al., 2019):

- **Signature-Based:** The IDS is trained with known attack scenarios and their signatures or patterns, which mean that the behaviour of the network can be compared against known signatures of adversaries to detect them. This is mostly for known attacks and will not be effective against new attacks that the model was not trained with.
- **Anomaly-Based:** Instead of looking at known attacks, this type of IDS will determine a baseline for normal behaviour in the network, such as the traffic, messages sent and the time intervals between communications, so that behaviour that deviates too much from this baseline can be deemed abnormal, being classified as a potential attack.

Detection systems are found to be effective, though depending on the algorithms implemented, the amount of overhead can be extremely high, as well as the fact that it is

possible to fool ML models, as discovered from adversarial examples. In addition to intrusion detection, it is also important to detect software vulnerabilities and the presence of malware, for which similar models can be employed (Dibaei et al., 2019).

(Sharma and Möller, 2018) look at the importance of protecting the in-vehicle network, especially the ECUs which govern different vehicle tasks, proposing an approach for intrusion detection and its subsequent prevention, known as IDPS. The in-vehicle bus is assumed by the authors to be the CAN bus. The IDS on its own simply monitors and detects potential attacks based on strange behaviour, whereas the IPS is the ability to respond to a potential intrusion, which then combines to form the IDPS. IDS either monitors internal activity, network activity, or even a mixture to detect unusual activity that could indicate an intruder. Detection is done either via signature (i.e., patterns) or behaviour, meaning unusual patterns or behaviours that may not be known, which can be useful especially for unknown or unpredictable attacks. IDPS uses neural networks and the relevant datasets to detect various attacks such as DoS and impersonation. IDPS in these cases used k-NN (Nearest Neighbour) classifiers, giving the following results:

1. For CAN traffic logs used as training datasets, 80% classification accuracy if the information was from a dataset, 68% if it was unknown (i.e., from outside the training data).
2. Use of SVM resulted in 82% accuracy for dataset information, and 59% accuracy when outside the datasets.

The authors proposed that recurrent neural networks (RNN) be used for the IDPS architecture, with past data being used to train the model so that it can predict future data packets, which works well especially due to limited vehicle data knowledge in the context of cybersecurity. The neural network predicts a packet and then compares it to the actual packet, constantly feeding this difference back into the system so it may learn, with big differences being considered anomalous. The IDPS can be considered a work in progress and should be able to detect in real time, with low error rates and low power consumption.

2.7.3 Intrusion Tolerance

Systems are considered intrusion tolerant if they remain available and retain security properties even if some of the components are compromised in an attack, typically using a protocol called Byzantine Fault Tolerance (BFT), which states that provided all of the components or replicas are not failed, the system will keep working, but in intrusion tolerance these failures/faults are assumed to be the work of an adversary (Bessani, 2011). The intrusion tolerance algorithm in its most basic form can be described in this way:

1. Let there be a value m which is a positive integer greater than or equal to 1.
2. For this value, the number of participants as per the Byzantine agreement will be $3m+1$ (e.g. for $m=1$ there will be 4 participants).
3. The functional property and system security will be maintained so long as an adversary does not overcome $m+1$ participants or more.
4. The participants, also known as replicas, will be cleansed periodically to move them out of their failed states through some predefined cleansing rate.

In order for intrusion tolerance to work properly, the following properties are important (Bessani, 2011):

1. The system must work well under all conditions, meaning it should be robust, so that even when under attack it keeps working well.
2. There must be diversity in security measures or otherwise, especially as common vulnerabilities are a weakness in the protocol.
3. The system must detect and cleanse intrusions to make sure that the adversary does not continue compromising replicas throughout the system until they eventually compromise them all. The attacks must be detected and stopped where possible.
4. Confidentiality, while not vital, could be a requirement in some cases.

By periodically rejuvenating multiple replicas of the same system or its software, it is possible to always keep parts of it available. If for a number of replicas, f out of $3f+1$ are compromised, then the system will still remain available, giving time to cleanse as well as remove any traces of an adversary (Platania et al., 2014). Replicas can be assigned the following states (Platania et al., 2014):

- **Correct:** The replica is working completely fine.
- **Compromised:** The replica is not behaving according to the algorithm, showing arbitrary behaviour.
- **Crashed:** The replica has stopped working and needs to be restarted.
- **Rejuvenating:** The replica is being reset, so compromised or not it will become a crashed replica and be reloaded.
- **Partitioned:** A replica that is considered separated from other replicas due to its inability to communicate with enough replicas.

Diversification of replicas allows for them to be defended across space, as they cannot all be vulnerable in the same way as their attack surface may alter, so there is no guarantee of compromising one and therefore being able to compromise another. However, with enough time, an adversary could still learn to compromise the diverse replicas, so the replicas should be periodically rejuvenated and diversified in such a way that they are not similar to any other replicas in the past or present, which is defence across time (Platania et al., 2014). For a system where it is hard to both know and mitigate all attacks while also maintaining availability, such as in CAVs, this could prove to be a good defence to model for cyber-resilience.

(Bessani, 2011) looks at being intrusion tolerant, compared to fault tolerance. While fault tolerance maintains availability, intrusion tolerance means security properties are maintained as well. Replication is already found to be effective both in terms of performance and minimal use of resources, but more work needs to be done on the effectiveness of diversifying replicas and making protocols more robust. The author makes it clear that open problems include the lack of definitive rules on how the system should react to intrusion, as well as making sure that state transfers in intrusion tolerance are time-bounded in a way that makes sure the adversary cannot delay the protocol to buy themselves more time to compromise a greater number of replicas.

(Sousa *et al.*, 2010) note that proactive recovery has been used to improve the resilience of intrusion tolerant systems in the past, making sure that the minimum replicas are present to keep the system available and working properly. The authors of the paper look at how proactive recovery could improve intrusion tolerance now. If faulty behaviour of a replica is

detectable, the replica will be recovered as soon as possible (which is important as the fault should be fixed as soon as possible), and the implementation of intrusion tolerance with proactive recovery was indeed able to increase resilience, being effective at dealing with DoS attacks both from inside and outside the network.

Cyberattacks are increasing, particularly against critical infrastructure. Therefore, (Platania *et al.*, 2014) present an intrusion tolerance system which defends across space and time, with diversification as well as proactive recovery. It uses the BFT protocol as is the case with all intrusion tolerance, and showed 95% confidence over a 30 year period for the intrusion tolerance model even for a 1TB state transfer (Platania *et al.*, 2014). The proactive recovery works for large states, with fast data retrieval and low bandwidth usage as the two strategies. Proactive recovery and diversification protect against attacks but not against flaws in the protocol itself such as those that make it vulnerable to more attacks. Replicas created of the software in parts of the system are diverse (i.e., similar but different, so no one attack works on all), and these replicas are either correctly working, compromised, crashed, or rejuvenating (i.e., being reset/cleansed). A replica could also be partitioned so they are separate from the others until they are working properly, as they cannot communicate with enough replicas in this state. The adversary was assumed to be immensely powerful with the ability to compromise replicas but not affect communications if they are working correctly, with no physical access to the system. It cannot bypass any implemented cryptographic methods either (Platania *et al.*, 2014).

(Popov, 2017) looks at modelling cyberattacks in software, with the idea that failure regions (i.e., states where software no longer works properly) are introduced when the software is compromised. The model architecture looks at fault tolerance, with different software that could be used to protect systems, with replicas being used to implement Byzantine fault tolerance (BFT) protocol. Methods such as patching and cleansing are to be used for the maintenance model of the software, with different attacks such as independent or synchronized attacks, and showing that reliability is affected by the different attack models. The authors model a 2-channel software system with different maintenance policies and uses a SAN model to create it as a network for simulation. For the 2-channels, 1-out-of-2 is used meaning that as long as one of the two channels is not compromised/failed then the system is available and can continue operation. The attacks in the model of the author are not ones that cause instant failure, but successful attacks that reduce reliability (i.e., advanced persistent threats). Cleansing is found to have one basic issue, which is that if a threat has previously compromised the software, even if it is removed, it is possible to do it again, even in a shorter amount of time. Patching can be costly, as well as potentially causing further issues if the patch is not done properly, so the better of the two should be chosen. Therefore, the author has chosen to compare cleansing and patching to see their effectiveness relative to one another. The experiments in question, created in Mobius 2.4 are as follows (Popov, 2017):

1. 2-channel system without maintenance (no cleansing or patching).
2. 2-channel system with cleansing only.
3. 2-channel system with patching only.
4. 2-channel system with cleansing and patching.

The simulations were set to be around 350 days at 10-day intervals, and calculation of the failure probability was done for 50-day intervals, repeating studies three hundred times. If the

required confidence level of 95% in the results was not reached, then repeats were done five hundred times instead. Cleansing and patching are first compared in the case of independent attacks, with maintenance rates of 0.01 (around 15 minutes). In the absence of maintenance, failure rate was shown to grow. For the first 250 days cleansing does better than patching, after which patching is better at reducing failure rate (Popov, 2017). When combined, as expected both forms of maintenance together rapidly reduce failure rate. Next, the author looks at the effects of synchronized vs. independent attacks, with independent attacks being more successful but less likely to cause failure than synchronized attacks. Both patching and cleansing alone seem to be less successful for synchronized attacks, with combined maintenance methods causing the system to behave as it would under independent attacks, which is preferable as this means at least one channel is kept available, meaning that patching and cleansing complement one another and should ideally be modelled together. By using deferred patching (i.e., making sure there is a problem before patching), solutions can then be implemented when needed rather than constantly applying patching. The results show that cleansing could be very effective in countering attacks, and patching depends on the vendors, so it cannot be adapted to system needs as the system designers cannot control this (Popov, 2017). Results suggest that deferred patching in fault tolerance could be better, but this requires more detailed exploration. Therefore, the system could work with a focus on cleansing while also having deferred patching. The author focused on fault tolerance but proposed extending the work to intrusion tolerance to assess its effectiveness against different attacks.

An architecture for intrusion-tolerance in Avs is proposed by (Vöelp and Esteves-Verissimo, 2018). The vehicle has diverse sensors and replicas, with sensor fusions and the pre-processing of visual data. This then goes into a capsule network which has both the complex manoeuvres, and the fail-safe linked into a safety kernel (for deciding how to perform a move) which allows for a rejuvenation process of all replicated components of the AV. The two vulnerable areas indicated here were the AI stack and the vision, suggesting that the perception and ML components of the vehicle are where it is best to apply intrusion tolerance. (Singh, Cukier and Sanders, 2003)

(Singh, Cukier and Sanders, 2003) state that intrusion tolerance is an emerging approach to allow systems to continue working when compromised by introducing replication. This paper attempted to validate such a security measure by creating SAN models to check the efficacy of such an idea before it can be deployed. A major source of uncertainty in security validation will always be the behaviour of the attacker as this is often unpredictable and an unknown that is hard to model depending on the type of attack. It is said that all models of intrusion tolerance need to include sub-models of the attacker, the intrusion tolerance mechanism as well as the system and its states. The SAN model used included one for a replica, a host, and the manager that controls the replicas (i.e. cleansing or otherwise). These are all joined together in a composed model which can allow different parts to be increased or decreased via the 'Rep' function. For each domain it was found that putting few hosts was a good idea, as intrusion tolerance in this case was very sensitive to how many domains were available to start new replicas.

New attacks are constantly being developed against systems with the aim of corrupting components and violating the integrity property of security as discussed in (Pal *et al.*, 2006). This has not been addressed much at the time of this paper, so the ITUA architecture has been

proposed which involves integrating multiple defence mechanisms into a security architecture with auto-adaptive intrusion tolerance to ensure as much security as possible. Intrusion tolerance itself aims to maintain operational capability when the system is under attack, especially as many of these attacks tend to result in corruption or loss of control of various resources (e.g. CPU cycles, memory, bandwidth, processes, and hosts).

The authors in (Pal *et al.*, 2006) created a prototype model to test for adaptive intrusion tolerance for the ITUA architecture to see its tolerance to attacks, with managers for the security and replicas each being contained within a relevant security domain which creates boundaries making it harder for an attacker to have direct access to each part of the system just by getting access to one area. Replicas are also started in a probabilistic way by managers to make sure a compromised manager in the system cannot take control of how replicas are placed or started, and all replicas have public keys which are known by all those who interact with it to ensure it is legitimate.

The ITUA architecture contains a rapid reaction loop where a quick response to attacks is made based on information from a sensor and IDS, with the loop monitoring the effectiveness of the response to the attack as well as the type of attack (e.g. same attack or multiple attacks, etc.). It is also possible to isolate hosts and domains by suspending trust based on observed behaviour, removing them from the architecture. It was found that auto-adaptive intrusion tolerance is possible, as the system can now adapt to an adversary and create various responses that the attacker must overcome, especially given that there were good response times by all parts of the system. This was said by the authors to be the best method of surviving attacks as it combines protection, detection and adaptation (Pal *et al.*, 2006).

The main contributions by (Pal *et al.*, 2006) involved an auto-adaptive intrusion tolerant architecture, and the information gained about it based off experience and testing (i.e. it was very effective). Techniques and mechanisms included the adaptive form of BFT replication, where corrupt replicas are removed and replaced with new ones, graceful degradation, host redundancy (i.e. extra hosts where needed) with a decentralized architecture, injecting uncertainty into responses and the integration of multiple defence mechanisms into a single architecture.

The identified gaps here are as follows:

1. The issue of authentication, so an adversary that sends false information could impact the system, meaning the safety implications need to be modelled.
2. How effective are defences like IDS and intrusion tolerance?
3. Is one kind of defence enough or do I need multiple at once?

2.8 Vehicle Standards

An important part of correctly building safe and secure CAVs is to have regulatory bodies as well as good standards for all manufacturers to abide by. Some of the most important ones so far are:

1. **ISO 26262:** The main standard for the functional safety of all vehicles, with guidelines and vocabulary.
2. **ISO 21448:** Safety of any function being incorporated into a vehicle, as each functionality has its own issues to consider.

3. **ISO 21434:** The main and most recent cybersecurity standard when it comes to vehicle design, including the guidelines, governance, and necessary threat analysis.
4. **SAE J3016:** The list of taxonomies and definitions in relation to the vehicle, driver as well as developing various regulations and specifications. This standard also contains all the levels of vehicle autonomy and their details.
5. **PAS 1885:** A standard for basic principles of vehicle cybersecurity, regarding how organizations should be dealing with cybersecurity in terms of understanding, culture, and the overall cyber-landscape.
6. **BS 31111:** Like PAS 1885 but created for managing cyber-risk and resilience for governing bodies and executive management.

Definitions and some descriptions are given by (Sanders and Meyer, 2001), describing SAN as a method of measuring performability used since the 1980s. The SAN can be used to look at features of a model such as performance and dependability, as well as being used to represent the behaviour of various systems with time constraints. These can be used as extensions to Petri nets and can be used to model a range of scenarios and systems including fault tolerance and system degradation, making them especially useful for modelling attacks and defences in my case. Effective models must have as little ambiguity as possible while having a sufficient level of abstraction (i.e., remaining sufficiently generic) to allow a system to be represented easily but also in a formal way to allow reliable results to be obtained. Activity networks can be considered generic Petri nets consisting of timed activities for state transition, places to describe states, as well as input and output gates that can check requirements or assign values, respectively. All input or output gates are connected to an activity, as well as to one unique place, but it is possible to connect different gates from an activity to a single place. When using SANs, all probabilistic behaviour (i.e., parameters and mathematics) must be made clear both in terms of variables and all equations relating to the statistical models of any activities. The timed activities of SANs start at activation and end at completion for the duration of the timed activity (i.e., the enabled period).

The NIST special paper by (Ross, McEvelley and Carrier Oren, 2016) consisted of different considerations for security, starting with fundamentals such as capabilities and dealing with various faults, as well as the strategy for security. The life cycle of the system must also be considered in terms of security, including management, design, analysis, and disposal (i.e., end of life cycle). An extremely useful element was the security policy, consisting of the confidentiality, integrity, and availability (CIA) triad of the different elements of the system.

BS 31111 (BSI, 2018a) looks at resilience and risk management, which requires adequate resources to meet the objectives in their policy. Security arrangements currently do not carefully consider factors that can adversely affect resilience, which can lead to projects being implemented that consider the benefits of the system without looking at current or emerging threats. A resilient organization must have strong governance, use of resilience on their designs and on the organizational level. There must also be continuous and strong cybersecurity, with the ability to learn and change from changing situations, while also sharing information effectively. It is also important that there is a cyber-aware culture, to influence human behaviour in a way to limit breaches due to human error which may be due to a lack of cybersecurity knowledge. There are many different rules and regulations that must be followed, including the ability to adapt, transforming business, risk management, and

correct response. Communication is key in all of these, as all the involved groups need to be able to make their positions and relevant issues known.

The ISO 26262 standard (BSI, 2018b) is intended for functional safety, namely for different devices produced, such as those in the automotive sector. It aims to create a safety culture within an organization, effectively promoting and communicating functional safety, while also managing the quality of system safety. The standard consists of vocabulary, management, phases (i.e., concept, where risks and hazards are determined), product development and safety analysis (i.e., ASIL safety levels for devices). There are also additional guidelines on how to apply the standard, not just for cars but for other vehicles such as motorcycles.

PAS 1885 (BSI, 2018c) is for how an organization should deal with venturing into the space of vehicle security, looking at resilience and risk management. Existing standards must be met, as well as looking at cyber risks, levels of preparation and prevention. It is also important to look at putting in the correct amounts of resources, with strong governance of cybersecurity and awareness of the changing cyber-landscape.

The purpose of the ISO 21434 (BSI, 2021) standard is to address cybersecurity in E/E systems and architecture. It is important to have cybersecurity policies and processes, manage risk and foster a culture of security within any organization that considers cybersecurity important. General considerations are the overall cybersecurity, such as what someone could do to try and cause problems, but in no way a specific idea, this is basic and more of a starting point. Cybersecurity should be regularly audited to check for compliance, with good planning and making sure all responsibilities are clear when it comes to security.

Cybersecurity must be regularly monitored, with regular evaluation for the life cycle of the product or system, with regular analysis and management for any vulnerabilities that emerge. Threat analysis and risk assessment methods include identifying threat scenarios, rating their impact (i.e., how dangerous), looking at the potential attack path taken by the adversary and the feasibility of the attack. After this the level of risk can be determined so an organization knows how seriously to treat a risk.

The code of practice for safety and cybersecurity by (IET, 2021) aims to not only hold organizations accountable for issues in these areas but to help them better manage these problems. The principle behind safety and security in digital systems which are connected to physical systems are that if it is not secure it cannot be considered safe, as there is no solid line between the safety and security of digital systems and control systems, meaning there is a lot of overlap and therefore a lot of safety implications. There is a lack of a current common method of approaching safety and security, with different groups of people such as IT experts or engineers seeing them in different ways and therefore placing priority on different areas, so it would help a lot if there was a consensus of what exactly is the most important when it comes to these two areas, especially when they overlap. Safety and security must be an active part of all processes, being taken into consideration when considering risks, while implementing methods to keep architectures resilient but with solutions being proportional to the level of risk. Considerations for safety and security need to be system-wide, with assessments performed to determine the best solutions to any risks.

Solutions to the possibility of faults occurring for a system and its components can be grouped as follows (IET, 2021):

1. **Fault avoidance:** Create the system with no faults in the first place so there is little to no future correction to be made.
2. **Fault removal:** Faults must be found and removed through the appropriate inspection methods that are of high quality.
3. **Fault tolerance:** The system should be designed in a resilient manner to deal with faults and remain available even when encountering some, avoiding any total failure.
4. **Fault warnings:** There must be methods of alert for faults as well as giving time for the system to recover properly from such issues.

It should be noted that this code of practice is intended for the engineers involved and those who manage these systems, so that they can use these good practices when making decisions regarding their systems.

In J3016, the vehicle standard aims to provide information into the recommended practice of a vehicle, covering the levels of autonomy for vehicles (SAE, 2021):

- **Level 0:** No driver assistance, meaning the driver is entirely responsible for the vehicle and its motion.
- **Level 1:** Driver assistance, where parts of the DDT (Dynamic Driving Task) are done by the vehicle, but the remainder must be performed by the driver, and the fall-back is still the driver.
- **Level 2:** There is partial driving automation, so the system performs motion control, but the driver is still the fall-back, with a limited ODD (Operational Design Domain, meaning the constraints under which the vehicle works correctly).
- **Level 3:** In conditional driving automation, the system is still in charge of the DDT, but now object and event detection and response are done by the system, with the driver being a fall-back from the DDT, with limited ODD.
- **Level 4:** High driving automation places all tasks in the hands of the system, with no driver fall-back, but the ODD is still limited, so it will only work in specific areas (i.e., with geofencing).
- **Level 5:** Full automation is the desired result, with the system doing everything without any driver input or exceptional circumstances, and no ODD limitation.

Driver automation is done by performing dynamic driving tasks (i.e., during motion, with changes) and with potential for object event detection and response (i.e., see an obstacle, respond by avoiding or stopping), and having a fall-back, such as in the case where the vehicle cannot handle it. The range of motion is also important, as it may be controlled both laterally and longitudinally, with some aspects given to the driver in lower levels of automation (SAE, 2021).

(BSI, 2022) makes it clear that safety is implemented at separate phases, such as in the design phase, verification phase, and validation phase, so that performance can be checked by different means including testing and technical reviews, and simulations. To improve safety of intended functionality, it is advised to have the full system specification, as well as the identification and evaluation of hazards (i.e., those caused by a functionality). Modifications should be made where needed to reduce risk, with verification and validation of safety. For safety in automated systems, I must make assumptions to effectively parametrize the model as much as possible, especially where I do not know all the information.

Here the authors of the IEEE standard state the minimum assumptions that need to be made about any automated system when developing a safety model. Some important terms include (IEEE Standards Association, 2022):

1. Acceptable risk: Sufficient level of safety, as there will always be some risk, but this needs to be within a certain bound to pose the least threat possible.
2. Event: Any changes to the driving system or its environment.
3. Formal models, which are designed mathematically or otherwise, for simulation, of the system in question to describe its behaviour.
4. Representation: All the information about a system in a particular case or scenario which can allow its effective description.
5. Safety envelope: The limits and conditions under which the system operates to ensure safety.
6. Safety-related Model: The model of the safety system to show all aspects of safety as well as the potential failings of such a system.

Any safety-relevant object such as other vehicles that could play a role in safety in the form of an event that could potentially be hazardous. Anyone who is not a vehicle user (i.e., a pedestrian or other non-driver) can be considered a vulnerable road user (VRU) in these cases. There are only some cases that assumptions can be made for, specifically scenarios such as (IEEE Standards Association, 2022):

1. Driving next to other road users.
2. Driving behind other road users.
3. Driving in between road users of varying speeds.
4. Interactions with VRUs on the road (i.e., how to deal with a pedestrian crossing).
5. Road users that may or may not be visible (i.e., obstructed by objects or poor weather).

In all these cases every element that is not the fixed road environment (i.e., scenery) is considered a dynamic entity and therefore any assumptions made are not always true as the environment with dynamic users will vary, generating hazards at any given time.

Assumptions made for modelling assume that the static environment will not change in any way, so my focus would be the dynamic users such as vehicles or pedestrians.

An example of an autonomous vehicle does not always have to be a car, as an existing example that may also be deployed is the driverless bus. In (Myklebust, Stålhane and Jenssen, 2023), a trust case is shown, as well as different security cases. The trust case here is where the bus is driverless, but its operator is moved to a remote-control room where they can take control if anything goes wrong. Another essential element is the standards for safety and security, one of the most recent and relevant being the ISO 21434:2021 which specifies the cybersecurity risk management requirements for vehicle electronic and electrical system throughout the entire vehicle life cycle. The standard includes requirements for manufacturers making components for parts of autonomous vehicles. Some AI standards are also required as a lot of ML algorithms are not always transparent so explaining them is difficult, meaning that there must be some sort of standards of the transparency of AI and what it does in CAVs.

Vehicle standards will not directly impact my model, but they are still important as the gap here is that all these either look at standards from a safety or security viewpoint, but not both.

my work addresses the issue by looking at both, so the results or methodologies could be used in future to help in developing standards that will no longer just consider a single aspect of vehicles.

2.9 Modelling

While not related to the background of CAV, this project will involve having to model the behaviour of a vehicle in the presence of various attacks to effectively evaluate cyber-resilience. As many behaviours and attacks against CAVs cannot be known or predicted fully, probabilistic modelling, simulations and potential other models will have to be used to get the closest approximation to the expected on-road behaviour of a CAV.

2.9.1 Stochastic Activity Network (SAN)

SAN models, also known as stochastic activity networks have been used in performance models for decades, even being used in software such as Mobius, which is currently being used in this project to model CAV systems to investigate their cyber-resilience, so that I can see how the system behaves under different circumstances (Sanders and Meyer, 2001).

The SAN model is said to consist of the following components for its ‘Atomic’ components, namely for a single network not yet joined to any others (Sanders and Meyer, 2001):

- **Place:** These are circular objects in the network that are assigned a value to indicate a state, such as compromised or failed. The system usually transitions between these when being simulated.
- **Timed Activity:** An activity that happens over time, set to perform a task at a fixed rate according to whatever distribution is chosen for it, such as exponential distribution.
- **Instantaneous Activity:** If an event happens on a state is reached connected to this, the activity happens immediately.
- **Input Gate:** This is a gate that the system passes through, with a predicate/requirement has based off existing states or parameters in the system which must be met to pass through this gate, otherwise the next state or event after this gate will not be reached.
- **Output Gate:** This is usually connected to another activity and values are output or set here before continuing to the other parts of the activity network.
- **Extended Place:** A more complex form of place which can be used for values such as arrays or structures, which are very good in a complex system.

These components are linked together via connections that can be curved or straight, with different cases occurring at activities, like if or case statements in programming. Various devices can be modelled this way, as well as being able to model Markov chains (Popov, 2015). Once the ‘Atomic’ component is created this can then be linked to other components, so that a place can exist in two different models, allowing another model, such as an attack, to affect another system. This can then be parameterized for numerous iterations and simulated either via numeric solver or Monte Carlo simulation.

2.9.2 ADVISE

One of the formalisms supported in Mobius is ADVISE, which allows for the creation of very specific attack models in the form of adversary profiles. This includes the following (Sanders and Applequist, 2012a):

1. **Attack Steps:** Actions taken by the adversary, with each step described by inputs, outputs, and the properties of the action itself.
2. **Access:** Areas the adversary has access to or may eventually gain access to. This consists of variables which store the access states, so it is known if the adversary has access to certain areas or not.
3. **Knowledge:** What kind of knowledge is available now or in future to an adversary, such as passwords to a certain area of the system which could allow for more effective attacks.
4. **Skill:** The capability of the adversary in different areas, which are stored as different variables indicating how strong the adversary is in different areas.
5. **Goal:** The aim and motivation of the adversary, which can range from stealing information to trying to shut down the entire system.

The ADVISE attack models are created and combined to form an attack execution graph, which can then be assigned different information such as the cost of an attack (i.e. the damage), preconditions, execution times (i.e. time intervals between attack steps), and the different outcomes from an adversary.

The authors in (Ford *et al.*, 2013) look at the implementation of ADVISE formalism in the Mobius tool. The formalism creates an attack model that contains all the adversary properties which can then be converted into a single executable event model. The attack model contains multiple state variables that define the access, skill, knowledge, and goals of the adversary in question. The state variables are assigned values as follows:

1. **Boolean:** The access, knowledge and goals are just flags, indicating if the attack does or does not have certain abilities (e.g. for access, does the adversary have root admin).
2. **Skills:** This is a measurement of proficiency, ranging from 0-1000.

The attacks steps are model elements that can cause the model to change state. These each contain a precondition, the relevant time distribution, and the cost of an attack. Each step is assigned its own outcomes which include the probability of the outcome, probability of detection and the probability of different effects.

The attack execution graph should not be confused with the adversary profile, which is what feeds the necessary parameters into the graph itself to define the adversary with key elements (i.e. the values of the skill, knowledge, etc.).

The attack model itself has a decision stage and an action stage. The decision stage looks at the different attack steps and outcomes that may happen in future and uses these to choose the most attractive next step for an adversary, and then the action is taken to perform the selected steps.

When using the solver for the state spaces to be simulated in ADVISE, it must be noted that not all models in Mobius can be transformed into Markov chains, which is especially true for ADVISE attack models as the adversarial decisions are not solely dependent on the current state. This is a limitation which makes this less suitable for modelling CAVs as Markov chains are a key element in the models and this not only removes this property in the state

space but also may overcomplicate the model itself with more information being required about the adversary that I simply may not know, while using simple SAN modelling allows for more general adversaries and a higher level of abstraction.

(Mariotti *et al.*, 2023) propose an extension to the already existing ADVISE formalism into a greater level of abstraction by modelling the attack patterns using the CAPEC (Common Attack Pattern Enumerations and Classifications) database and adversaries profile using the TAL (Threat Agent Library) database.

The entries used from the CAPEC database describes an attack pattern as well as the following elements relevant to an attack:

1. Description of the attack.
2. Prerequisites to carry out the attack.
3. Resources required (e.g. devices and tools).
4. Skills required for the attack.
5. Consequences of the attack, including the scope and impact.

The goal of ADVISE Meta is to make it easier to model by removing the need to do so much modelling work by hand, consisting of the following parameters:

1. Component: Elements that could be part of the system.
2. Relationship: Could exist between components in the system.
3. Attribute: Characteristics of components which can be used as parameters for attack steps connected to any component.
4. Access: Level of access an adversary has at the start of the attack.
5. Skill: Capability of the adversary.
6. Knowledge: Information possessed by the adversary, in advance that may affect the attack.
7. Other State Variable(s): Other variables that could define adversary goals.
8. Attack Step: Defines steps of an attack that could be performed by an adversary.
9. Adversary: Defines the profile of an adversary with various characteristics.
10. Metric: Those such as goals are metrics, but more can be added.

The ADVISE Meta modelling involves adding all the elements into a system diagram and setting all the parameters and connecting components through their relationships based on system architecture.

The attack steps consist of the attack cost, execution time, preconditions, success probability, and detection probability.

The TAL database contains the following elements:

1. Access

2. Intent
3. Limits
4. Objective
5. Outcome
6. Resources
7. Skill Level
8. Visibility

It is possible to combine elements from both CAPEC and TAL with the ADVISE formalism to create the ADVISE Meta. Remaining work includes trying to integrate defences into the models, with current defences represented by the probability of a successful attack or other components such as the strength of existing defensive measures.

2.9.3 GSPN

In GSPN and Interval GSPN (IGSPN), the timed activities are all exponentially distributed regarding their firing times. GSPN expresses models as some of the following (Galdino and Maciel, 2006):

1. Finite set of places (i.e. how many states I can go to).
2. Finite set of transitions (i.e. the arcs or timed activities).
3. Inputs, outputs, and inhibitor functions.
4. Priority function to determine the importance of different transitions.
5. A function to associate the transitions with real values including the exponential parameters and the weights.

(Galdino and Maciel, 2006) investigated extending GSPN using an interval-based approach to create IGSPN. The aim of GSPN is to evaluate performance and reliability of petri net models, with IGSPN taking a closer look at the effects of variations in the exponential weights and transition rates. These are assessed by looking at various sources of error such as rounding off or truncation relative to the initial data, giving intervals that indicate the minima and maxima for each of these values.

What IGSPN does is that it adds an interval model to all these components of the GSPN, representing uncertainties that could exist in different model parameters. This suggests that while GSPN could be effective for reliable modelling, especially with the interval addition that could account for uncertainty, it is still restricted by the fact that all transitions are exponentially distributed, which limits the kind of behaviour that could be modelled only to that of events which are known to be exponentially distributed.

2.9.4 PRISM

PRISM is modelling method which represents different models using its own unique code. The latest version at the time of writing introduced the quantitative checking of probabilistic timed automata as well as probabilistic, nondeterministic, and real-time characteristics.

PRISM provides model checking for various probabilistic models, including discrete and continuous-time Markov chains and decision processes, as well as providing a language to represent the models with. It can be used in a range of applications including wireless technology and quantum cryptography.

Both deterministic and nondeterministic models are supported for both discrete and continuous time, with the ability to reuse models and can be described as follows (Kwiatkowska, Norman and Parker, 2011):

1. Discrete-time Markov chains
2. Markov Decision Processes (MDP)
3. Probabilistic Automata
4. Continuous-time Markov chains
5. Probabilistic Timed Automata (PTA)
6. Priced Probabilistic Timed Automata

The overall functionality of PRISM includes the ability to model and build different models including probabilistic timed automata (priced and regular), with the ability to add costs and rewards to enhance them further. There are also different model checking methods which can be done with symbolic and explicit-state, with discrete-time simulation and the ability to import models.

Probabilistic timed automata are a form of state machine with a real value clock (variables with real values that increase over time) added to them, to make them like timed automata with probabilistic choices which can allow them to behave like Markov decision processes. The transitions in these automata can reset the clock when the state changes, with the transitions being probabilistic and non-deterministic. If the costs and rewards are accounted for in these models, the automata can become priced probabilistic timed automata.

Probabilistic timed automata are verified in the following ways:

1. Quantitative abstraction refinement uses probabilistic abstractions and analysis to refine at each step of the model
2. Digital clocks are integer valued and perform automatic translation of the model to its equivalent finite state automata and checks it using Markov decision process-based methods.

Additionally, the discrete-event simulation engine can be used to debug and check models, and optimal adversary generation is also provided based on the reward models, creating counterexamples to the model in the form of attacks.

Such a modelling tool has many possibilities, especially with all the verification methods available, but it is all written as a coding language with very little visualization, making it harder to see how it may reflect the architecture of the system.

2.9.5 Other

Despite SAN being the key model, and the project being focused on this type of modelling, there are still other models that need to be considered as these are usually used to evaluate risks and threats to various parts of systems or processes. Some of these models include:

STRIDE: Looking at Spoofing, Tampering, Repudiation, Information Disclosure, DoS, and Elevation of Privilege. These are key security elements as they embody what an attacker may aim to achieve, so by assessing these it is possible to see which properties are at risk of being violated for specific system parts (Fowler, 2020).

DREAD: Developed by Microsoft, this looks at how much damage an attack can do, how easily it is reproduced, the extent to which it can be exploited (i.e., difficulty), users affected by the attack and how easy it would be to discover the vulnerability in question (Fowler, 2020). Each part is given a score, and the average of these scores is taken to evaluate the overall risk.

PASTA: The Process for Attack Simulation and Threat Analysis consider threats from the business and technical perspectives. This approach is centred around an attacker and aims to product output that evaluates and enumerates threat scores (Fowler, 2020).

HEAVENS: Short for ‘Healing Vulnerabilities to Enhance Software Security & Safety’, this is a systematic approach of modelling, which aims to determine the security requirements as well as performing security testing, it uses case studies, threat analysis and risk assessment, which can then be used to better identify, classify and assess threats and/or risks (Fowler, 2020).

EVITA: E-Safety Vehicle Intrusion Protected Applications evaluates protection of security engineering, threats and security requirements, with the original European project having focused on protecting the in-vehicle network in the automotive domain (Fowler, 2020).

OCTAVE: Operationally Critical Threat, Asset and Vulnerability Evaluation assessment looks at information security, identifying the threat, the assets and evaluating the vulnerability, so that the assets most important to an organization can be better protected by seeing which threats or vulnerabilities put these assets in danger (Fowler, 2020).

FMEA: Failure Mode & Effect Analysis looks at individual functions or processes, the failure that has occurred and performs analysis of the effect. This includes the frequency, severity, and detection of such a failure, taking note of recommended actions as well as who should be taking these actions to prevent such failures in the future (Forrest, 2010).

FMVEA: This is much like FMEA except that vulnerabilities are now also considered (hence the ‘V’), meaning that threats and their effects are considered in parallel with failures when evaluating the function or process in question. Threats and their agents are analysed repeatedly until there are none left, so this model now considers both safety and security (Schmittner, Ma and Smith, 2014; Schmittner *et al.*, 2015).

CHASSIS: Looking at assessing the potential harm to an information system, this also combined safety and security, both in terms of functional and safety/security requirements, while utilizing use cases. This type of model looks at users, functions, services, potential harms (as well as their scenarios), and mitigations that exist against harm in the current system (Schmittner *et al.*, 2015).

HAZOP: Hazard and Operability Study is for highly detailed system designs, ideally with multidisciplinary team that select a stage or section of the system/process, looking at each section like FMEA, so all hazards, implications and countermeasures are known (Crawley and Tyler, 2015).

Another note on models, specifically semi-Markov can be found in (Littlewood, 1979) looks at a modular program designed to transfer control between different modules. This is assumed to follow a semi-Markov process and is considered failure-prone with its processes being assumed to be Poisson for failures. The failure process of the program in question is said to be an asymptotic Poisson process. It is stated that two non-overlapping intervals of time could be considered asymptotically independent with their own blocks of behaviour. The contribution of this paper is therefore demonstrating that the time to an absorbing state for a semi-Markov model is asymptotically Exponential where the rate of an accident (i.e. absorbing state) is smaller than the rate of transition to the non-absorbing states, even being able to provide a formula for how this works.

For more details on any of these modelling types and how they work, the details can be found in Appendix 12.1 where they are fully explained with diagrams to show their processes where possible.

This section allowed for the evaluation of my choice of model for this work, which was as follows:

- ADVISE assumes I can model the attacks perfectly, not accounting for future attacks that may be completely unpredictable.
- GSPN is not flexible enough to have different parts that can be joined together (i.e. no separate attack or defence models).
- PRISM is entirely programming based with no visuals to map out the CAV model, just being a programming language.
- The other modelling methods are primarily for evaluation rather than any solid quantification, giving no ability to effectively simulate. Where simulation is possible, it is too complex and does not allow for effective application of parameters at the desired level of abstraction.

From this, I found that the most effective and desirable means of modelling was the SAN formalism as it granted flexibility, good visualization, and the ability to program my components in detail where desired.

3 Research Methodology

In this section, I will be going through my research objectives and methodology used to meet them for the project. The research itself is oriented at investigating the cyber-resilience of CAVs and requires stochastic modelling, meaning that it will require different simulation software and use of numerical methods.

3.1 Key Objectives

For the research, my objectives are as follows:

1. Investigation into the cyber-resilience of CAVs in the presence of different attacks and defences.
2. Implications of different cyber-attacks on the safety of CAVs.
3. Modelling CAVs in the presence of different hazards, as well as attacks that could affect how the vehicle responds.
4. Creating models and framework for different defences that could then be implemented in the model to see their capabilities and effects on vehicular safety.
5. Validation of CAV models developed as well as their chosen parameters, which can be justified as well as being reproducible.

3.2 Methodology

The method of the research will be based on sufficient knowledge of CAV systems and subsystems to create good stochastic models, which will evaluate the resilience of vehicular components as well as the vehicle itself overall.

The actual models used the SAN formalism, with successful attacks increasing the risk of systems not working properly and leading to an accident which is the absorbing state for the model, like in (Popov, 2017). Any parameters obtained for the models such as rate of hazards occurring are either obtained from existing papers such as ITSC (Popov *et al.*, 2022) or determined within a plausible range through the use of sensitivity analysis, in which case their sensitivity analysis will be present in the respective results section of the model in question.

3.2.1 The Möbius Tool

As vehicles are complex systems, an effective modelling tool is needed that can encompass all the distinctive features and model them in a straightforward way. For this, I chose the Möbius tool due to its flexibility, especially its compositionality, and various inbuilt distributions, which will make stochastic modelling much easier, especially due to inbuilt simulators (i.e., Monte Carlo, Transient Solvers, etc.) which could be incredibly difficult to write and program from scratch (Sanders and Applequist, 2012b).

The tool allows for a model to be created, with the following sections for the overall package, which can then be archived and stored to be sent to other computers (Sanders and Applequist, 2012b):

1. **Atomic:** This is where models such as SAN are created, as state machines which used both timed and instantaneous activities, with places to show the current state. They

can reflect various parts of the system, being incomplete but with the potential to link them together.

2. **Composed:** This is where different elements can be linked together, meaning that states can be shared, which is integral to modelling a system under attack, as the state indicating the system has been compromised will come from an attack model and need to be reflected in the system or component model. Aside from joining, replication is possible, which can be useful if the intention is to have more than one of the same components being simulated at once.
3. **Reward:** This indicates various features such as what results and conditions are going to be observed, for which section of the models (i.e., individual SAN or composed), and the duration of simulation.
4. **Study:** This is where different ‘Experiments’ are placed, with variables being altered according to what is going to be tested and observed.
5. **Transformer:** Based on the model so far, as well as the number of potential states, this is where a transformer such as a state space generator will be run to create all states for simulation if they are required.
6. **Solver:** This is the final stage for simulation, which will use the chosen transformer (i.e., state space) and simulate the model depending on user choice (i.e., Monte Carlo or Transient Solver).

Timed activities offer a range of distributions, as well as different cases based on probabilities of a particular event being successful or not. In the case of this project, the focus of these activities will be modelling the rates of different events using exponential distributions. Any other details of the model and my specific experiments are described at later points in more detail specific to the work.

I used the software to create SAN models of the vehicle, initially for the whole vehicle network and its core components, with each one being subject to an attack. These are generic to start with, but the aim will be to have realistic values introduced, as well as introduction of actual safety critical systems (i.e., safety monitors and hazard perception) where I can see how much safety is affected by attacks.

3.2.2 Vehicle Simulators

While modelling CAVs stochastically to evaluate their behaviour is useful and allows room for analysis numerically, this still does not give an entirely accurate representation of the implications on the road when vehicle security is compromised. To achieve this, there needs to be a way to translate it into physical events, meaning that the CAV must be simulated as close to reality as possible.

There are numerous car simulators, but a well-known one that has been discussed and proposed is the Car Learning to Act (CARLA) simulator, which allows for various vehicles, scenarios, and environments, including changing the weather. This simulator uses various programming languages, with Python being one of the main ones.

There will inevitably be challenges in simulating the numerical results, by applying constraints to the simulation, but it is key to seeing the effects on CAVs. Vehicle simulators often need to run for prolonged periods of time and require dedicated computers, meaning that it may be necessary to communicate with a research team or company that already makes use of the desired simulator on its own computers.

3.2.3 Hazard Models

The next stage will be to have a vehicle that is exposed to different hazards with and without attacks, as hazards are where safety issues are mostly likely to occur. I created a hazard model with different parameters including those for attacks, as well as parameters that will either scale the probability of accidents up or down depending on what they affect and where.

By changing these different parameters and simulating them, I can now obtain results that to plot to observe graphs of the time to an accident or the probability of an accident over time as different parameters are activated. Aside from changing these scaling parameters and those related to the attack models, I also hope to introduce models for existing defences, as it is important to see how different defences help or hinder the cyber-resilience of CAVs, the core of the research.

Through repeated testing and observation, I was then be able to see the effects of the different parameters on system safety, as well as which areas of the vehicle are most affected in terms of safety and security. From this, more detailed models of the vulnerable sections can be created specifically to test their cyber-resilience with or without attacks and defences.

3.2.4 Attack Models

The attack models used are broad and generic to encompass all the different possible attacks, consisting of the following elements:

1. Attack Intensity: This is the rate at which attacks occur over a period (e.g. once per hour).
2. Success Probability: The probability that an adversary will succeed in executing an attack against an element of the vehicle.
3. Deterioration: This is the failure modes, where an attack can cause deterioration in the functionality of the component it has attacked by some amount (e.g. safety monitor becoming 10 times worse under attack).

This is a suitable form of attack modelling as (Cao *et al.*, 2022) is an example of a perception attack against the LiDAR, using spoofed signals at close range to effectively 'delete' a pedestrian from the vehicle view, with 92.7% success. This means that by being confined to the period where the vehicle is close to the adversary can define the attack intensity, the success as a percentage can define the success probability and the effect of the attack is the level of deterioration.

The attack models used have a high level of abstraction to account for all possibilities without the need to constantly create new and specific attacks, especially given that not all attack types will be known (e.g. zero-day vulnerabilities), so this level of abstraction can cover both known and unknown adversaries. It is possible to create extremely specific models of the known attacks to date, but the benefits of such studies would be limited as the vulnerabilities from known attacks are usually quickly patched making their safety implications less valuable. The focus of my work is to offer a methodology for studying the safety implications of cyber-attacks on CAVs irrespective of the mechanisms used to execute attacks which may or may not be known.

3.2.5 Model Validation

My models so far can be considered valid, having consulted with my supervisors as part of a best effort way of validation which made use of all sources and methods of testing possible given the available resources. The experimental design stage was assumed to be unnecessary at this stage, as many of the experimental features were built into Mobius, meaning that more sophisticated methods of designing an experiment were not used at this point. Experiments at this point essentially consist of many fixed parameters, with the relevant ones being toggled between 1.0 (due to the use of indices in applying vulnerability modes to the system) and a value defined in advance (i.e., by how much a certain attack could increase a risk) so that the effect of their presence can be observed for different experiments.

In terms of parameters, they are divided into the known and unknown. The known parameters can be obtained as realistic values from studies on vehicle behaviour, which in this case already existed in (Popov *et al.*, 2022) which is the base of the model which is used, with the key difference being the addition of an attack model, so that the safety can be assessed when cyber-attacks on different elements of the vehicle are present, both individually and in combination. The unknown parameters had to be obtained via sensitivity analysis, where a range of plausible values were tested, shrinking the range gradually until I reached an acceptable parameter value for each unknown, such as with the rate of attack or recovery rates of failed components.

4 Concept of Modelling

Initially, the work was started by creating a model that was designed to assess vulnerability modes, which was then used to assess the impact of different vulnerability modes on safety in the form of the risk of an accident. This was published in DESSERT'2022 and later improved to better reflect a real model, after which it was then partitioned into vehicle 'sections' to model a vehicle with a CAN bus and ECUs, with attacks being system wide.

It must be noted that the model is rather abstract, but the later partitioning of the model in which functionalities are grouped (i.e. failure modes for perception and safety monitor) means that there is some architecture present where the partitions can represent ECUs or domains which when attacked will impact the vulnerability/failure modes. The architectures from (Dibaei *et al.*, 2019) can reflect each of these parts, with the initial model being closer to gateway-based, while the partitioning can be either domains or zones, but does not necessarily represent a main or central vehicle computer. The usefulness of the model depends on what I am looking for, which in this case is the risk to safety and where it is coming from, so it can be prevented in future vehicle designs.

The simulations in the experiments were initially conducted with Monte Carlo, but this gave inaccurate results as well as having very lengthy simulation times compared to a numeric solver, so all my results where Mobius was used were obtained using a numeric transient solver due to less time and greater accuracy of results generated.

4.1 SAN

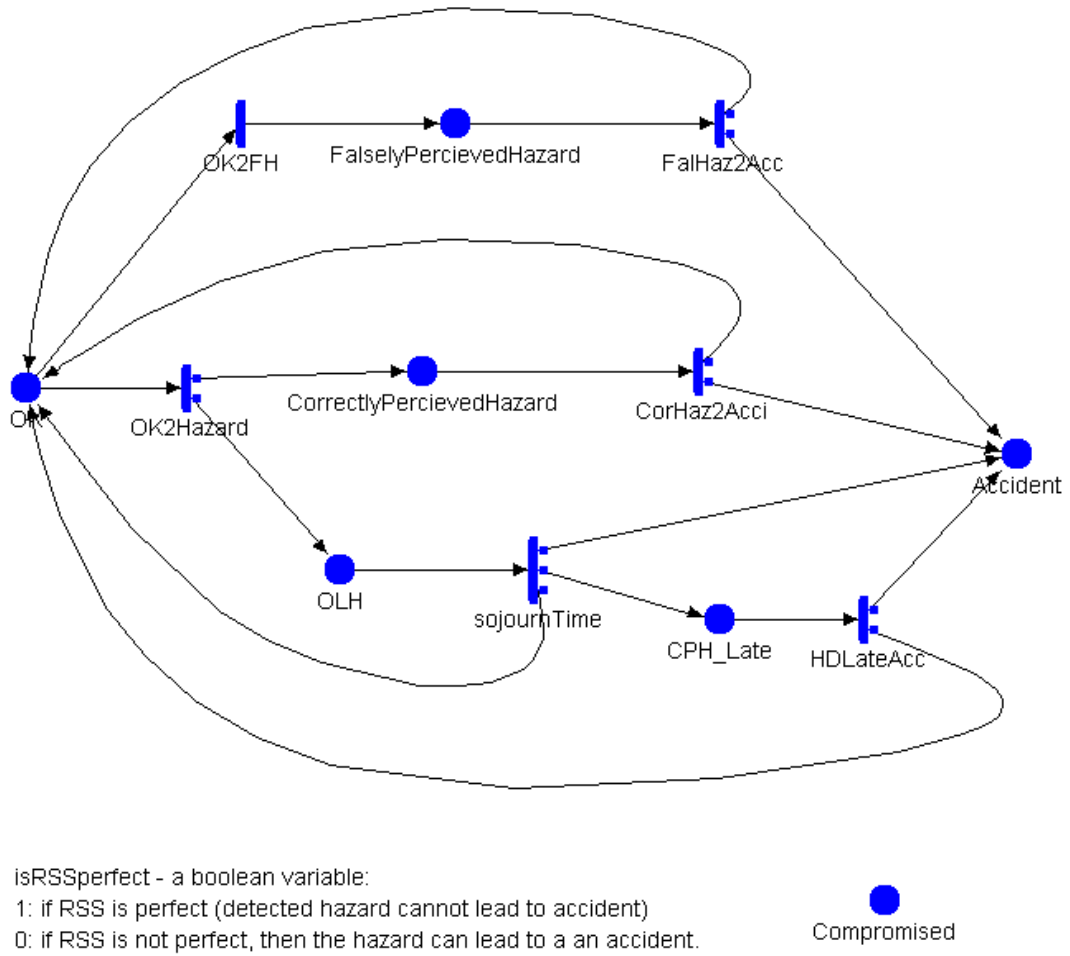


Figure 6: Hazard Perception State Machine

Looking at my models, I see in Figure 6 the structure of the hazard perception model as a state machine, where the system moves from state to state, with the use of timed activities that have been given their own probability distributions to control the rate at which events happen.

This model has several paths depending on the way hazards are perceived, and if dealt with in a timely manner then the system can still return to an 'OK' or safe state but will go to the absorbing 'Accident' state otherwise, which means that the hazard was not dealt with in a timely manner by the perception or the safety monitor.

The system consists of the following states (Kosari, Popov and Roy, 2022):

- **OK:** This state models road conditions free from road hazards.
- **Accident:** The vehicle has now had an accident, which is the absorbing state.
- **FalselyPerceivedHazard:** This state is for when the vehicle incorrectly perceives the current road conditions as hazardous.
- **CorrectlyPerceivedHazard:** This state models the hazardous state on the road which the vehicle has correctly perceived.

- **OLH:** This state models a hazardous situation on the road overlooked by the vehicle.
- **CPH_Late:** This state is for a road hazard which is eventually detected by the vehicle but with some delay. The vehicle has remained unaware of the hazard for some time at this point.
- **Compromised:** This state captures the fact that a cyber-attack on AV has succeeded and is linked to the attack model, which can affect hazard perception and safety monitors through their relevant vulnerability modes.

Additionally, timed activities are used to model events such as the separate ways of perceiving hazards at different rates, using exponential distributions for all timed activities so far. The list of these activities consists of (Kosari, Popov and Roy, 2022):

- **OK2FH:** This timed activity captures the intervals between falsely perceived hazards.
- **OK2Hazard:** This time activity models the intervals between hazardous situations on the road.
- **CorHaz2Acci:** This timed activity models the duration of a road hazard which is correctly detected on time. Either it escalates to an accident, or instead the hazard “goes away” (i.e., the situation on the road is not dangerous any longer).
- **FailHaz2Acc:** This is the duration of the false hazard and can have outcomes like CorHaz2Acci – either the false hazard goes away without any visible consequences, or the false hazard escalates to an accident.
- **sojournTime:** Models the duration of overlooking a hazard, given the hazard has occurred. This interval can end in one of the following possibilities: Escalation to “Accident” moving to “OLH”, when the hazard is ongoing and is eventually detected but with some delay or returning to the “OK” state.
- **HDLateAcc:** This transition models the duration of hazard after its successful detection (with a delay), which may result in an accident or going back to the “OK” state.

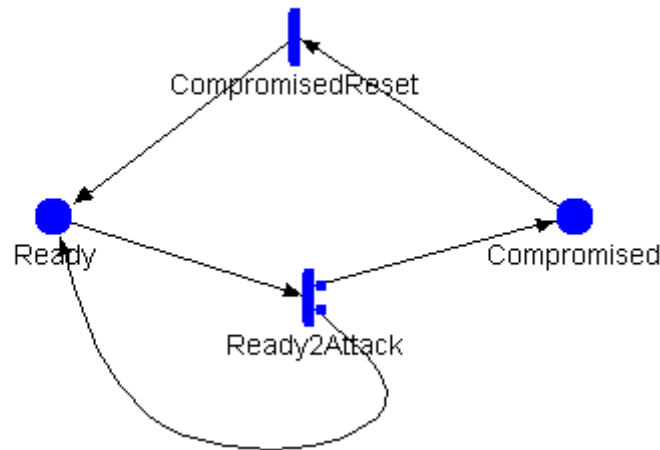


Figure 7: CAV Attack Model

In Figure 7, there is a generic attack model where the attacker is assumed to be able to constantly perform attacks at a fixed rate, and compromised the system at a certain rate

provided the attack is successful, after which the attack will reset and be performed again. The following states describe the model:

- **Ready:** An attacker is ready to perform an attack on the hazard perception system.
- **Ready2Attack:** This is the time interval between attacks being carried out, which can either be successful, or the attack will be repeated continuously.
- **Compromised:** The attack has succeeded and whichever system it is linked to can now be considered compromised, as this causes any active vulnerability modes to impact the system and increase risk of accidents. These can be found across different timed activities within areas such as hazard perception and the safety monitor.
- **CompromisedReset:** The interval between periodically reloading an attack and moving out of the ‘Compromised’ state.

This new attack component is my initial contribution to the model, as the initial model comes from (Popov et al., 2022) which looks at vehicle safety in the presence of road hazards. Now, I have improved the model to account for attacks affecting different components of the hazard perception model to meet my goal of investigation safety implications of cyber-attacks on CAVs. This allows me to model and get results that show the safety implications of cyber-attacks against CAVs.

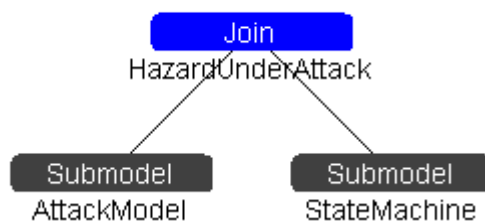


Figure 8: Hazard and Attack Model Join

In the above figure, I can see that the attack model and the state machine (i.e., hazard perception + safety monitor) are joined so the compromised state can be shared for the attack to have some effect on the vehicle model to be able to investigate it.

Now, based on potential vulnerabilities that could result in the failure of hazard perception and safety monitors, the following parameters were devised to be varied to investigate the effects of a cyber-attack on the vehicle (Kosari, Popov and Roy, 2022):

- **HazardScale:** This parameter allows us to study how sensitive vehicle safety is to deterioration of hazard detection rate caused by an adversary.
- **DelayScale:** This parameter allows us to look at how sensitive vehicle safety is to increased delays in perceiving road hazards.
- **AccidentScale:** This parameter allows us to study how compromising integrity of the safety monitor rules can impact safety.
- **OH2CH_scale:** This parameter allows us to assess how sensitive vehicle safety is to attacks that cause hazards to be overlooked.
- **LateCH2AccScale:** This parameter is like AccidentScale but looks at vehicle sensitivity to those difficult road hazards which are overlooked but correctly detected later.

Parameter	Value	Function
CH2AccRate	1.0E-4	Accident rate of correctly perceived hazard [hours ⁻¹]
CH2OKRate	856.3	Rate of recovery from correctly perceived hazard [hours ⁻¹]
FH2AccRate	1.0E-5	Rate of accident due to falsely perceived hazard [hours ⁻¹]
FH2OKRate	856.3	Rate of recovery from falsely perceived hazard [hours ⁻¹]
HazardRate	197.4	Rate of hazard occurrence [hours ⁻¹]
MH2AccRate	0.001	Accident rate due to delayed hazard perception [hours ⁻¹]
MH2CHRate	2250.0	Rate of an overlooked hazard being correctly perceived with delay [hours ⁻¹]
OH2OKRate	1000.0	Overlooked hazard recovery [hours ⁻¹]
attackRate	0.1	Rate of attack [hours ⁻¹]
attackReset	1.0E-4	Rate of attack model reset [hours ⁻¹]
attackSuccess	0.5	Rate of successful attack [hours ⁻¹]
falseHazardProb	0.0	Rate of false hazards [hours ⁻¹]
isRSSperfect	0	Is the safety system perfect (1) or not (0)?
missHazardProb	0.05	Rate of missing/overlooking a hazard [hours ⁻¹]

Table 5: Table of Fixed Parameters

In Table 5 these are the fixed parameters that I have used to begin with, initially not taking false hazards into account. These are used throughout the experiments in conjunction with the different vulnerability mode parameters to see their effects individually and in combination with one another. The definitions of these parameters do not change throughout the experiments, and all timed activities use exponential distributions. Additionally, due to its greater accuracy, the method of simulation involved using a transient solver with an accuracy of 10^{-9} as well as a period of 100-9100 hours, with increments of 1000.

4.2 Reward Model

The SAN model which is then linked through a composed to the attack must be assigned a reward model. The aim of this is to check for when the ‘Accident’ state is reached and ensure that I have a return on the average amount of times this absorbing state is reached so that the probability of accidents can be measured to evaluate safety. This can be seen in the table below which models the reward as a performance variable which checks the absorbing state in the SAN model.

Performance Variable : TimeToAccident		
Affecting Models	StateMachine	
Impulse Functions		
Reward Function	<i>(Reward is over all Available Models)</i> if(StateMachine->Accident->Mark() > 0) return(1); else return(0);	
Simulator Statistics	Type	Instant of Time
	Options	Estimate Mean
		Estimate Variance
		Include Lower Bound on Interval Estimate
		Include Upper Bound on Interval Estimate
		Estimate Distribution
		Estimate out of Range Probabilities
		Confidence Level is Relative
	Parameters	Start Time
	Distributions	Lower Bound on Distribution Estimate
		Upper Bound on Distribution Estimate
		Step Size on Distribution Estimate
	Confidence	Confidence Level
		Confidence Interval

Table 6: Accident Reward Model

The reward model is then linked to the study which is where the relevant parameters are held (e.g. rate and probability for timed activities), with the entire model using Exponential distribution for all its activities.

4.3 Attack Model

The attack model, as specified before, is Exponentially distributed, simply returning a defined rate of attacked and its relevant probabilities of success, which will be the main type of model used unless specifically stated otherwise for a particular model and is shown below.

Timed Activity:	Ready2Attack
Distribution Parameters	Rate return(attackRate);
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 return(attackSuccess); case 2 return(1.0 - attackSuccess);

Table 7: Attack Model

4.4 DESSERT'2022

The results from the DESSERT'2022 paper by (Kosari, Popov and Roy, 2022) indicate that individually there is very little effect on safety from attacking a vulnerability mode on its own. However, attacking all vulnerability modes at once shows a massive increase in the risk, indicating the importance of exploring combinations of attacks on vulnerability modes as well as making the model more efficient and realistic to make sure that these results are as close to reality as they can be, using known parameters that can be looked up as well as sensitivity analysis where needed. False hazards were also not accounted for here, so that branch of the model was inactive, so this was also later explored. The model was simulated across multiple points in time from 100-9100 hours in 1000-hour steps, using a numeric solver with a 10^{-9} precision, as the risk of an accident evolves over time when observing CAVs. Additional details can be found in Appendix 13.1 with a slightly refined version of the model which is used later.

4.5 Combination Testing – Improved Model

Following on from the DESSERT'2022 model, it was concluded that not only is it worth testing combinations of vulnerability modes, but a more robust model which is not prone to any potential numerical issues, so that I can now begin to test combinations of vulnerability modes as opposed to single modes, which proved to have very little effect, with delayed hazard (i.e. DelayScale) being the only one to have any noticeable value above the baseline. The models were simulated across multiple points in time from 100-9100 hours in 1000-hour steps, using a numeric solver with a 10^{-9} precision, as the risk of an accident evolves over time when observing CAVs. The full model documentation can be found in Appendix 13.2. The vulnerability modes are the components at this stage and can represent a gateway architecture as they are very loosely grouped.

4.5.1 Main Model

The model parameter shown in Table 8 below. Many are the same as in the model above, except all combination of vulnerability mode is tested. Values now alternate between 1.0 (basically zero effect) and the chosen value of the vulnerability mode when activating (the

greater this is, the greater the effect by orders of magnitude), as well as changes to the attack model parameters and false hazard probability, which was not included before.

Parameter	Value(s)
AccidentScale	[1.0, 10.0]
CH2AccRate	1.0E-4
CH2OKRate	856.3
DelayScale	[1.0, 1000.0]
FH2AccRate	1.0E-5
FH2OKRate	856.3
HazardRate	197.4
HazardScale	[1.0, 10.0]
LateCH2AccScale	[1.0, 10.0]
MH2AccRate	0.001
MH2CHRate	2250.0
OH2CH scale	[1.0, 10.0]
attackRate	0.1
attackReset	2.0
attackSuccess	0.2
falseHazardProb	1.0E-4
isRSSperfect	0
missHazardProb	0.05

Table 8: CAV Model Parameters (Improved)

The values in Table 8 are used as the rates of events and probabilities in the timed activities that make up the SAN model. Apart from the attack parameters the rest were obtained from (Popov et al., 2022). The vulnerability modes were assigned in a way to increase the probability of a safety failure by at least one order of magnitude, while the attacks were determined through sensitivity analysis.

From this and the number of combinations, it is clear there are thirty-two total experiments, so I can now see what combinations are most dangerous to the CAV when it comes to vulnerability modes.

Experiment	AccidentScale	DelayScale	HazardScale	LateCH2AccScale	OH2CH scale
Baseline	1.0	1.0	1.0	1.0	1.0
AccidentScale	10.0	1.0	1.0	1.0	1.0
DelayScale	1.0	1000.0	1.0	1.0	1.0
HazardScale	1.0	1.0	10.0	1.0	1.0
LateCH2AccScale	1.0	1.0	1.0	10.0	1.0
OH2CH	1.0	1.0	1.0	1.0	10.0
AccDel	10.0	1000.0	1.0	1.0	1.0
AccHaz	10.0	1.0	10.0	1.0	1.0
AccLate	10.0	1.0	1.0	10.0	1.0
AccOH2CH	10.0	1.0	1.0	1.0	10.0
DelHaz	1.0	1000.0	10.0	1.0	1.0
DelLate	1.0	1000.0	1.0	10.0	1.0
DelOH2CH	1.0	1000.0	1.0	1.0	10.0
HazLate	1.0	1.0	10.0	10.0	1.0
HazOH2CH	1.0	1.0	10.0	1.0	10.0
LateOH2CH	1.0	1.0	1.0	10.0	10.0
AccDelHaz	10.0	1000.0	10.0	1.0	1.0
AccDelLate	10.0	1000.0	1.0	10.0	1.0
AccDelOH	10.0	1000.0	1.0	1.0	10.0
AccHazLate	10.0	1.0	10.0	10.0	1.0
AccHazOH	10.0	1.0	10.0	1.0	10.0
AccLateOH	10.0	1.0	1.0	10.0	10.0
DelHazLate	1.0	1000.0	10.0	10.0	1.0
DelHazOH	1.0	1000.0	10.0	1.0	10.0
DelLateOH	1.0	1000.0	1.0	10.0	10.0
HazLateOH	1.0	1.0	10.0	10.0	10.0
AccDelHazLate	10.0	1000.0	10.0	10.0	1.0
AccDelHazOH	10.0	1000.0	10.0	1.0	10.0
AccDelLateOH	10.0	1000.0	1.0	10.0	10.0
AccHazLateOH	10.0	1.0	10.0	10.0	10.0
DelHazLateOH	1.0	1000.0	10.0	10.0	10.0
All	10.0	1000.0	10.0	10.0	10.0

Table 9: CAV Mode Combinations

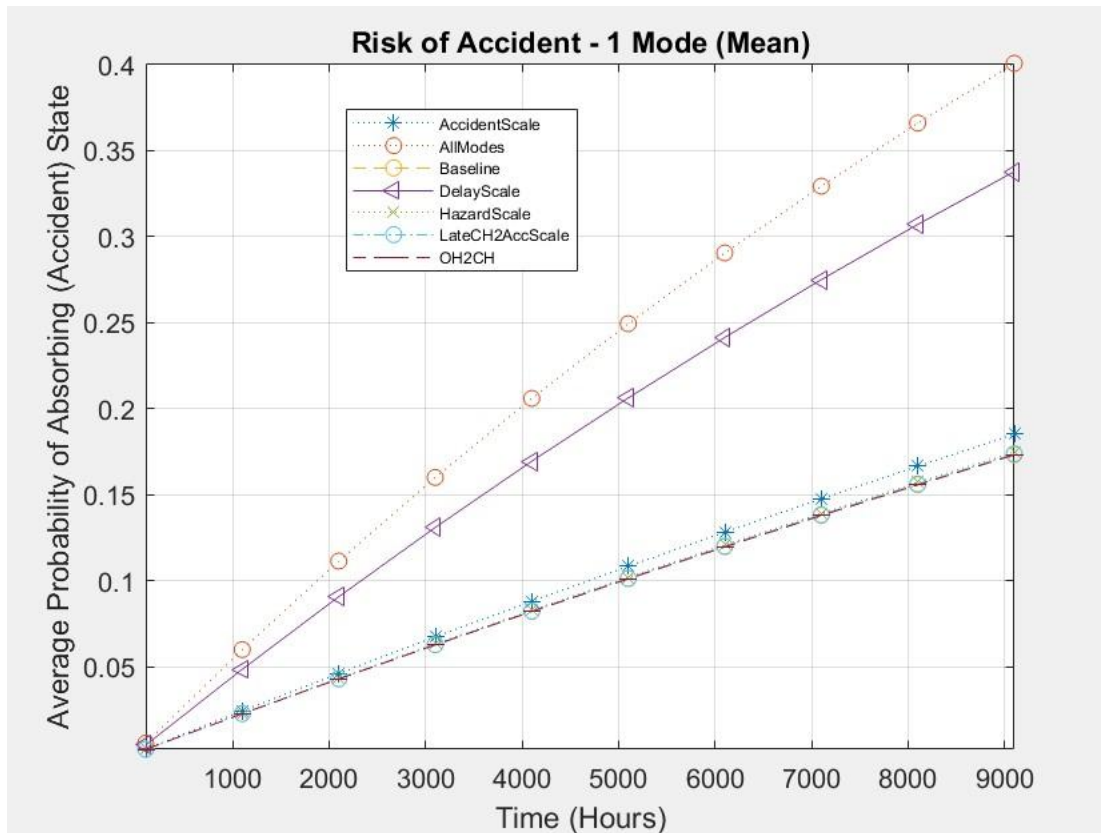


Figure 9: Single Vulnerability Mode Attack

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
AccidentScale	0.0022	0.0245	0.0462	0.0674	0.0881	0.1084	0.1283	0.1477	0.1667	0.1852
DelayScale	0.0045	0.0486	0.0907	0.1310	0.1695	0.2063	0.2414	0.2750	0.3071	0.3378
HazardScale	0.0021	0.0230	0.0434	0.0634	0.0829	0.1021	0.1209	0.1392	0.1572	0.1748
LateCH2AccScale	0.0021	0.0228	0.0431	0.0630	0.0824	0.1015	0.1201	0.1384	0.1563	0.1738
OH2CH	0.0021	0.0227	0.0429	0.0627	0.0821	0.1011	0.1197	0.1379	0.1557	0.1732
AllModes	0.0056	0.0600	0.1114	0.1601	0.2060	0.2495	0.2905	0.3294	0.3661	0.4008

Table 10: Single Vulnerability Mode Data

For the single mode vulnerabilities in Figure 9 and Table 10, they are ranked from most to least dangerous in terms of safety:

1. DelayScale
2. AccidentScale
3. HazardScale
4. LateCH2AccScale
5. OH2CH

From these the most dangerous vulnerability mode is DelayScale as this increases the probability of missing hazards or identifying them later. This is still much lower than compromising all vulnerability modes at once, suggesting that it is worth testing combinations of attacks to find the most dangerous combinations against CAVs.

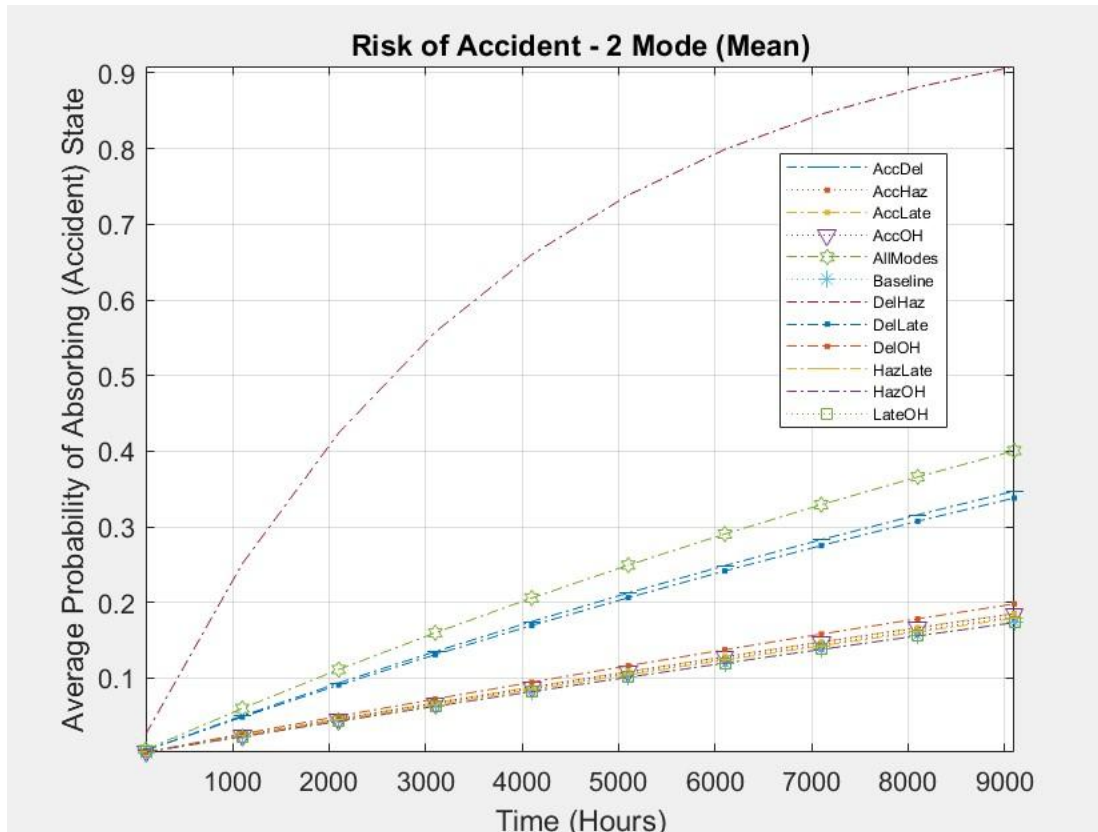


Figure 10: Two Vulnerability Modes Attacked

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
AccDel	0.0047	0.0503	0.0938	0.1353	0.1749	0.2127	0.2487	0.2831	0.3160	0.3473
AccHaz	0.0022	0.0239	0.0451	0.0658	0.0861	0.1059	0.1254	0.1443	0.1629	0.1811
AccLate	0.0023	0.0245	0.0463	0.0676	0.0884	0.1087	0.1286	0.1480	0.1670	0.1856
AccOH	0.0024	0.0244	0.0461	0.0673	0.0881	0.1084	0.1282	0.1476	0.1655	0.1851
DelHaz	0.0258	0.2512	0.4244	0.5575	0.6599	0.7386	0.7990	0.8455	0.8813	0.9087
DelLate	0.0045	0.0487	0.0908	0.1312	0.1697	0.2065	0.2417	0.2753	0.3075	0.3382
DelOH	0.0024	0.0264	0.0497	0.0725	0.0947	0.1165	0.1377	0.1583	0.1785	0.1982
HazLate	0.0022	0.0236	0.0445	0.0650	0.0851	0.1047	0.1240	0.1428	0.1612	0.1792
HazOH	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1199	0.1381	0.1559	0.1734
LateOH	0.0021	0.0228	0.0431	0.0630	0.0824	0.1015	0.1201	0.1384	0.1563	0.1738
AllModes	0.0056	0.0600	0.1114	0.1601	0.2060	0.2495	0.2905	0.3294	0.3661	0.4008

Table 11: Two Vulnerability Mode Data

For two vulnerability modes, going from most to least dangerous, the combinations are as follows:

1. DelHaz (DelayScale + HazardScale)
2. AccDel (AccidentScale + DelayScale)
3. DelLate (DelayScale + LateCH2AccScale)
4. DelOH (DelayScale + OH2CH_scale)
5. AccLate (AccidentScale + LateCH2AccScale)
6. AccHaz (AccidentScale + HazardScale)
7. AccOH (AccidentScale + OH2CH_scale)
8. HazLate (HazardScale + LateCH2AccScale)
9. LateOH (LateCH2AccScale + OH2CH_scale)
10. HazOH (HazardScale + OH2CH_scale)

In this case, as seen in Figure 10, the combination of DelayScale and HazardScale has greater effect on system safety than all vulnerability modes triggered together, suggesting that some combinations, such as those involving OH2CH may in fact make it less dangerous. This observation is quite surprising but was found to be due to how it is only present in the probability of the ‘sojournTime’ timed activity, mostly as the denominator in the probability calculations, which means its activation can lower probabilities in the branches here which can make accidents less likely in the model.

The presence of the DelayScale vulnerability mode in the attack is shown to be key to causing safety issues which follows on from the results from a single vulnerability mode attack. Now, I moved onto experimenting with triplet combinations of vulnerability modes to observe the safety implications here.

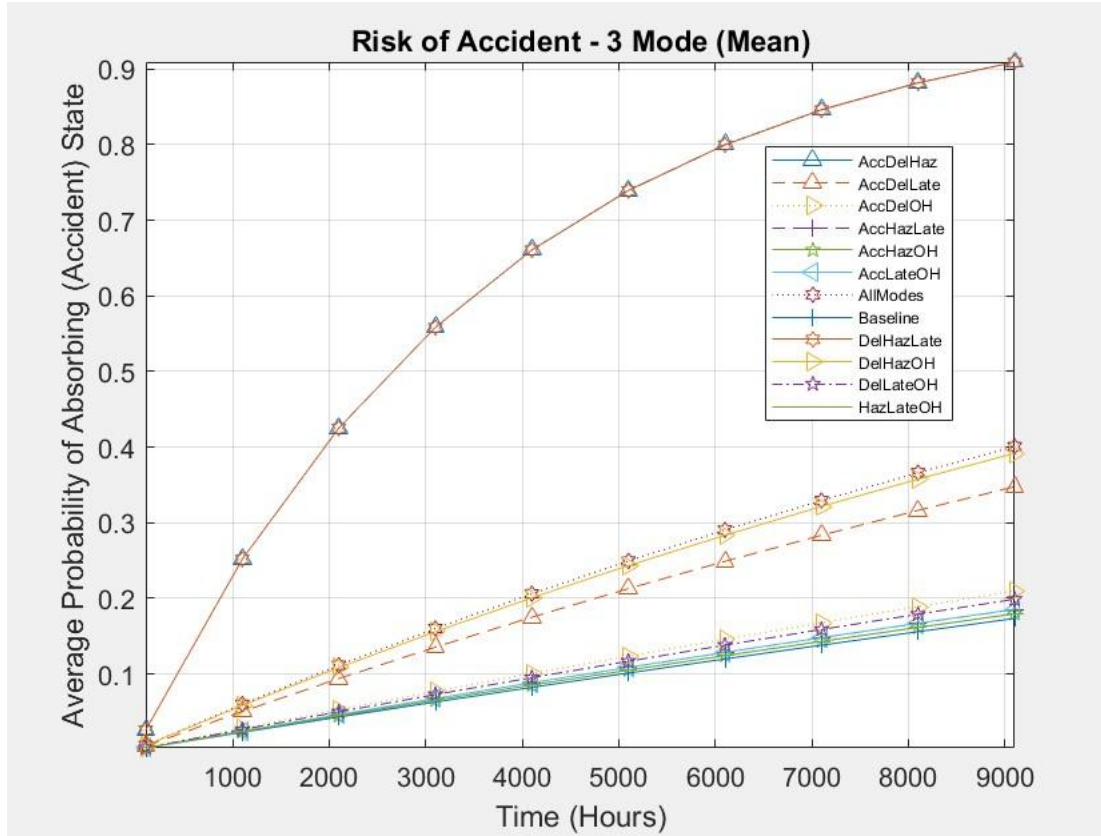


Figure 11: Three Vulnerability Modes Attacked

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
AccDelHaz	0.0259	0.2519	0.4254	0.5587	0.6610	0.7397	0.8000	0.8464	0.8820	0.9094
AccDelLate	0.0047	0.0503	0.0939	0.1354	0.1751	0.2129	0.2490	0.2834	0.3163	0.3476
AccDelOH	0.0026	0.0280	0.0529	0.0770	0.1006	0.1236	0.1460	0.1678	0.1890	0.2097
AccHazLate	0.0023	0.0245	0.0462	0.0675	0.0883	0.1086	0.1284	0.1479	0.1668	0.1854
AccHazOH	0.0022	0.0237	0.0447	0.0652	0.0854	0.1051	0.1243	0.1432	0.1616	0.1797
AccLateOH	0.0023	0.0245	0.0463	0.0676	0.0884	0.1087	0.1286	0.1481	0.1671	0.1856
DelHazLate	0.0259	0.2516	0.4251	0.5583	0.6607	0.7393	0.7997	0.8461	0.8818	0.9092
DelHazOH	0.0054	0.0583	0.1084	0.1558	0.2007	0.2432	0.2835	0.3216	0.3576	0.3918
DelLateOH	0.0024	0.0264	0.0499	0.0727	0.0950	0.1168	0.1381	0.1588	0.1791	0.1988
HazLateOH	0.0022	0.0236	0.0446	0.0651	0.0852	0.1049	0.1241	0.1429	0.1614	0.1794
AllModes	0.0056	0.0600	0.1114	0.1601	0.2060	0.2495	0.2905	0.3294	0.3661	0.4008

Table 12: Three Vulnerability Mode Data

For three vulnerability modes seen in Figure 11 and Table 12, the risk to safety from most to least is as follows:

1. AccDelHaz (AccidentScale + DelayScale + HazardScale)
2. DelHazLate (DelayScale + HazardScale + LateCH2AccScale)
3. DelHazOH (DelayScale + HazardScale + OH2CH_scale)
4. AccDelLate (AccidentScale + DelayScale + LateCH2AccScale)
5. AccDelOH (AccidentScale + DelayScale + OH2CH_scale)
6. DelLateOH (DelayScale + LateCH2AccScale + OH2CH_scale)
7. AccLateOH (AccidentScale + LateCH2AccScale + OH2CH_scale)
8. AccHazLate (AccidentScale + HazardScale + LateCH2AccScale)
9. AccHazOH (AccidentScale + HazardScale + OH2CH_scale)
10. HazLateOH (HazardScale + LateCH2AccScale + OH2CH_scale)

Once again, the combination of AccidentScale, DelayScale and HazardScale is the most serious, even more than combining all other modes together, and the presence of DelayScale with HazardScale is the key here suggesting that attacks on hazard perception are the key to serious safety issues.

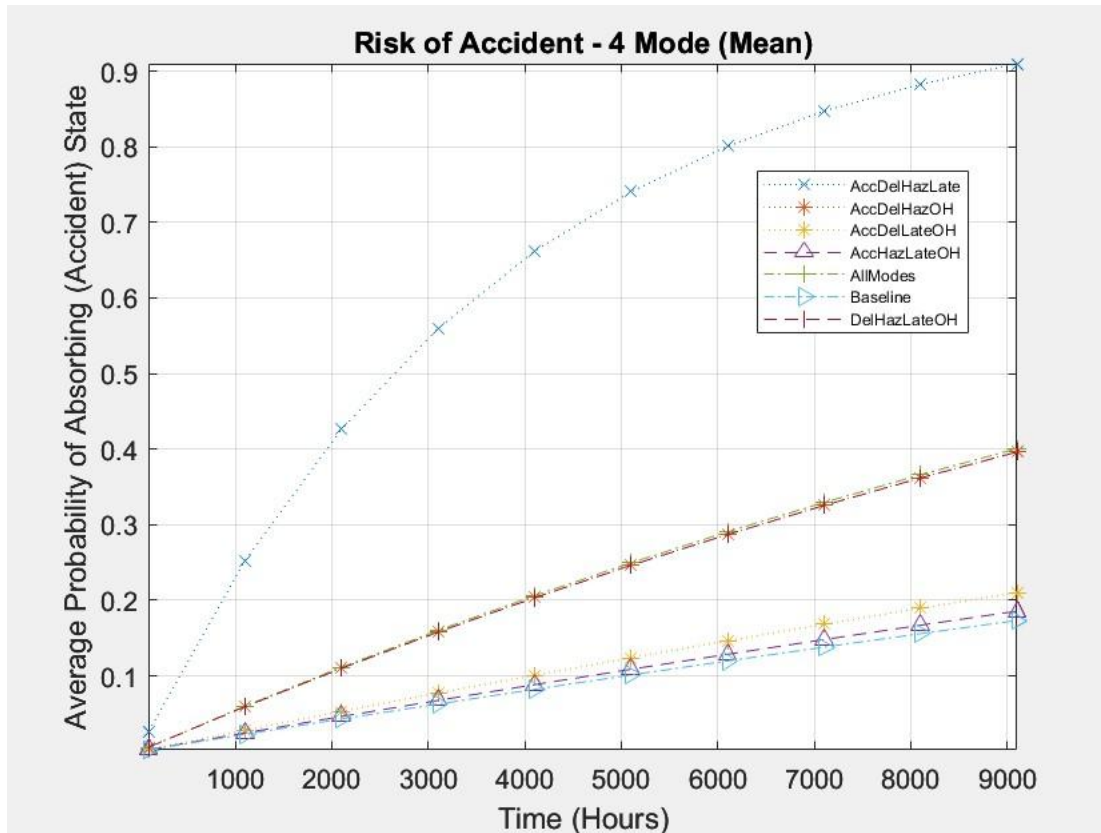


Figure 12: Four Vulnerability Modes Attacked

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
AccDelHazLate	0.0260	0.2523	0.4261	0.5594	0.6618	0.7404	0.8007	0.8470	0.8826	0.9099
AccDelHazOH	0.0055	0.0592	0.1100	0.1580	0.2034	0.2464	0.2871	0.3256	0.3260	0.3964
AccDelLateOH	0.0026	0.0281	0.0530	0.0773	0.1009	0.1239	0.1464	0.1682	0.1895	0.2103
AccHazLateOH	0.0023	0.0245	0.0463	0.0676	0.0884	0.1087	0.1286	0.1480	0.1670	0.1856
DelHazLateOH	0.0055	0.0591	0.1099	0.1579	0.2033	0.2463	0.2869	0.3254	0.3618	0.3962
AllModes	0.0056	0.0600	0.1114	0.1601	0.2060	0.2495	0.2905	0.3294	0.3661	0.4008

Table 13: Four Vulnerability Mode Data

From the data in Table 13 and Figure 12, the most to least dangerous combinations of four modes are as follows:

1. AccDelHazLate (AccidentScale + DelayScale + HazardScale + LateCH2AccScale)
2. AccDelHazOH (AccidentScale + DelayScale + HazardScale + OH2CH_scale)
3. DelHazLateOH (DelayScale + HazardScale + LateCH2AccScale + OH2CH_scale)
4. AccDelLateOH (AccidentScale + DelayScale + LateCH2AccScale + OH2CH_scale)
5. DelHazLateOH (DelayScale + HazardScale + LateCH2AccScale + OH2CH_scale)

From the four combinations, the presence of DelayScale, HazardScale and AccidentScale are modes which cause serious impact. When combined with LateCH2AccScale, this has the most serious effects, but when OH2CH is added this is instead weaker, which is supported by

the fact that attacking all modes gives lower risk compared to combinations where this is not the case.

4.5.1.1 Discussion

From this section of results, the following is clear:

- For single modes of attack, attacks on the 'DelayScale' vulnerability mode are most serious, as they interfere with the timely detection of hazards which reduces the window of time to react to the hazard. Therefore, this suggests attacks against individual perception components are quite serious.
- For two vulnerability modes, the combination of 'DelayScale' and 'HazardScale' is the most significant, as the vehicle now not only has difficulty reacting to hazards on time, but the number of hazards not being perceived means that there are many more hazards being detected late or not at all which can increase the risk of accidents. This indicates that the perception components are extremely vulnerable especially in the presence of hazards.
- For three vulnerability modes under attack, 'AccidentScale', 'DelayScale' and 'HazardScale' have the greatest safety implications. This is because there is now weaker hazard detection as well as late hazard detected coupled with the compromising of the safety monitor integrity through 'AccidentScale'. This means that the vehicle is not only dealing with hazards poorly, but the safety policies are unable to act to ensure safe driving throughout all this. Here, if the perception is compromised along with a safety monitor component, it only gets worse as the safety monitor is now weakened at the same time as perception so the CAV will be driving dangerously.
- For the four vulnerability modes, it is the combination of 'AccidentScale', 'HazardScale', 'DelayScale' and 'LateCH2AccScale' that causes the greatest risk to safety, as this brings the effects of the greatest risk for three modes and adds a new element of 'LateCH2AccScale' meaning that overlooked hazards may be less likely to be detected by the system later. The risk only gets worse as more components are compromised, and the last line(s) of defence to deal with hazards at the last minute are also weakened.
- It was found that 'OH2CH_scale' has little effect towards safety, causing the combinations that include it to be less dangerous (e.g. in the case of 'DelayScale' + 'HazardScale', where this combination had more risk than compromising all modes). This is due to it only existing within the 'sojournTime' timed activity (i.e. the area dealing with later hazards) at the same time as 'DelayScale' but primarily as a denominator of the probability calculations in the code which can cause it to lower the probability of success in different branches. This strongly suggests that for combinations of attacks, overlooking hazards must exist in them as part of the attack, so only some of the attack combinations are valid, indicating the need to refine the model further.
- Generally, activating all modes can still be considered a realistic worst case, but it seems that the modes on their own do not fully reflect a real CAV so an architecture-based model of similar design should be used in the next stages.

- There is only a single absorbing state, so the results are limited to one type of accident, when there could be many different scenarios where a safety failure is not necessarily catastrophic, limiting the extent to which the model can assess safety.
- A limitation of the model is that it assumes adversaries that occur at a fixed rate with some success probability, which is typically not 100% meaning that the model does not account for an adversary that can perfectly execute an attack every time without issue (i.e. I assume a limited level of expertise from the adversary).
- It must also be noted that while the model is limited to Exponentially distributed timed activities only, this is not a major issue as according to the findings of (Littlewood, 1979), all Markov models of this type are asymptotically Exponential irrespective of the distribution used meaning that my results for the risk would give an Exponential result over time anyway.

4.5.2 Realistic Attack

Continuing from the sensitivity analysis, the parameters used for attack were acquired from research on compromising SSH (Ramsbrock, Berthier and Cukier, 2007), which is not from the AV domain as these could not be found.

In this case, it is based off attacks from 229 IP addresses against four computers. The number of attacks for 24 days was 269,262 meaning that Attack Rate = $269262/229/4/24/24 = 0.51$, which is approximately 1 attack every 2 hours. The cleansing rate is chosen by sensitivity analysis, assuming attack in this case will always impact the system, using the attack model parameters from the table below. Any additional graphs can be found in Appendix 14.1.

Parameter	Value(s)
attackRate	0.51
attackReset	2.0
attackSuccess	0.0031

Table 14: Realistic Attack Parameters – Vulnerability Modes

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
AccidentScale	0.0021	0.0229	0.0432	0.0632	0.0827	0.1018	0.1205	0.1388	0.1567	0.1743
DelayScale	0.0023	0.0248	0.0469	0.0684	0.0895	0.1101	0.1302	0.1499	0.1691	0.1879
HazardScale	0.0021	0.0228	0.0430	0.0628	0.0823	0.1013	0.1199	0.1381	0.1560	0.1735
LateCH2AccScale	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1381	0.1559	0.1734
OH2CH	0.0021	0.0227	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1733
AllModes	0.0024	0.0258	0.0486	0.0710	0.0927	0.1140	0.1348	0.1551	0.1749	0.1943

Table 15: Realistic Risk of Accident - 1 Mode (Raw Data)

From the results in Table 15, the system is attacked for a single mode, which shows the following risk from highest to lowest for the modes:

1. DelayScale
2. AccidentScale
3. HazardScale
4. LateCH2AccScale
5. OH2CH

From this it can be determined that attacks which cause delays to the hazard perception (i.e. more often missing the hazards) have the most serious effects individually, while the least effect is from OH2CH_scale, so correcting a missed hazard is the least concerning in an attack on the CAV.

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
AccDel	0.0023	0.0250	0.0471	0.0688	0.0900	0.1106	0.1309	0.1506	0.1699	0.1888
AccHaz	0.0021	0.0228	0.0431	0.0630	0.0825	0.1016	0.1203	0.1385	0.1564	0.1740
AccLate	0.0021	0.0229	0.0432	0.0632	0.0827	0.1018	0.1205	0.1388	0.1568	0.1743
AccOH	0.0021	0.0229	0.0432	0.0632	0.0827	0.1018	0.1205	0.1388	0.1567	0.1743
DelHaz	0.0040	0.0433	0.0810	0.1173	0.1521	0.1855	0.2176	0.2485	0.2781	0.3066
DelLate	0.0023	0.0248	0.0469	0.0684	0.0895	0.1101	0.1302	0.1499	0.1691	0.1879
DelOH	0.0021	0.0230	0.0435	0.0636	0.0832	0.1024	0.1212	0.1397	0.1577	0.1754
HazLate	0.0021	0.0228	0.0431	0.0630	0.0824	0.1015	0.1201	0.1384	0.1563	0.1738
HazOH	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
LateOH	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1381	0.1559	0.1734
AllModes	0.0024	0.0258	0.0486	0.0710	0.0927	0.1140	0.1348	0.1551	0.1749	0.1943

Table 16: Realistic Risk of Accident - 2 Modes (Raw Data)

For two vulnerability modes combined, the results in Table 16, the most serious vulnerability modes that affect safety due to attacks are as follows (highest to lowest):

1. DelHaz (DelayScale + HazardScale)
2. AccDel (AccidentScale + DelayScale)
3. DelLate (DelayScale + LateCH2AccScale)
4. DelOH (DelayScale + OH2CH_scale)
5. AccLate (AccidentScale + LateCH2AccScale)
6. AccOH (AccidentScale + OH2CH_scale)
7. AccLate (AccidentScale + LateCH2AccScale)
8. AccHaz (AccidentScale + HazardScale)
9. HazLate (HazardScale + LateCH2AccScale)
10. HazOH (HazardScale + OH2CH_scale) and LateOH (LateCH2AccScale + OH2CH_scale)

This continues from the previous conclusion where DelayScale caused the most risk to safety, so combinations including it are more serious here, especially DelHaz (DelayScale and HazardScale), to the point where it is more dangerous than compromising all the modes together, which suggests that certain combinations of modes will negate the effects, specifically the presence of OH2CH_scale.

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
AccDelHaz	0.0040	0.0434	0.0811	0.1174	0.1523	0.1858	0.2179	0.2488	0.2785	0.3070
AccDelLate	0.0023	0.0250	0.0471	0.0688	0.0900	0.1107	0.1309	0.1506	0.1700	0.1888
AccDelOH	0.0021	0.0232	0.0438	0.0639	0.0837	0.1030	0.1219	0.1404	0.1586	0.1763
AccHazLate	0.0021	0.0229	0.0432	0.0632	0.0827	0.1018	0.1205	0.1388	0.1568	0.1743
AccHazOH	0.0021	0.0228	0.0431	0.0630	0.0825	0.1015	0.1202	0.1384	0.1563	0.1739
AccLateOH	0.0021	0.0229	0.0432	0.0632	0.0827	0.1018	0.1205	0.1388	0.1568	0.1743
DelHazLate	0.0040	0.0433	0.0811	0.1174	0.1522	0.1857	0.2178	0.2487	0.2784	0.3069
DelHazOH	0.0024	0.0256	0.0484	0.0706	0.0923	0.1134	0.1341	0.1543	0.1741	0.1933
DelLateOH	0.0021	0.0230	0.0435	0.0636	0.0832	0.1025	0.1213	0.1397	0.1577	0.1754
HazLateOH	0.0021	0.0228	0.0431	0.0630	0.0824	0.1015	0.1202	0.1384	0.1563	0.1738
AllModes	0.0024	0.0258	0.0486	0.0710	0.0927	0.1140	0.1348	0.1551	0.1749	0.1943

Table 17: Realistic Risk of Accident - 3 Modes (Raw Data)

For three modes attacked at once, as seen in Table 17, the risk to safety can be listed as followed in descending order:

1. AccDelHaz
2. DelHazLate
3. DelHazOH
4. AccDelLate

5. AccDelOH
6. DelLateOH
7. AccHazLate and AccLateOH
8. AccHazOH
9. HazLateOH

Here, once again, the combinations including DelayScale and later HazardScale have the most effective risk to safety when attacked. For the three modes, adding AccidentScale has the greatest effect, whereas those with OH2CH_scale tend to present much less risk.

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
AccDelHazLate	0.0040	0.0434	0.0812	0.1176	0.1524	0.1860	0.2182	0.2491	0.2788	0.3073
AccDelHazOH	0.0024	0.0257	0.0485	0.0708	0.0925	0.1137	0.1345	0.1547	0.1745	0.1938
AccDelLateOH	0.0021	0.0232	0.0438	0.0640	0.0837	0.1030	0.1220	0.1405	0.1586	0.1764
AccHazLateOH	0.0021	0.0229	0.0432	0.0632	0.0827	0.1018	0.1205	0.1388	0.1568	0.1743
DelHazLateOH	0.0024	0.0257	0.0485	0.0708	0.0925	0.1137	0.1345	0.1547	0.1745	0.1938
AllModes	0.0024	0.0258	0.0486	0.0710	0.0927	0.1140	0.1348	0.1551	0.1749	0.1943

Table 18: Realistic Risk of Accident - 4 Modes (Raw Data)

Finally, the combination of four vulnerability modes can be observed in Table 18, the combinations from most to least risk of accident are as follows:

1. AccDelHazLate
2. AccDelHazOH and DelHazLateOH
3. AccDelLateOH
4. AccHazLateOH

The highest risk comes from the attacks where HazardScale and DelayScale are present, continuing from before. When AccidentScale is added, as seen in three vulnerability combinations, this continues to be the highest risk. For four modes, the presence of LateCH2AccScale continues to keep it high, whereas if OH2CH_scale is used, the risk is lower, which has been consistently true throughout this.

Using the set of parameters derived from (Popov *et al.*, 2022) confirmed the earlier observations. Therefore, it can be concluded that the DelayScale parameter is the highest risk on its own, then HazardScale can be added in combination. After this, AccidentScale can be added for three modes and then anything other than OH2CH_scale in combination can have the highest risk. It can be determined that while there is no change in the danger of combinations relative to one another, the fact that the realistic attack model has a lower success probability means that the risks in general are lower for this type of model.

4.5.2.1 Discussion

From this section of results, the following is clear:

- For single modes of attack, attacks on the ‘DelayScale’ vulnerability mode are most serious, as they interfere with the timely detection of hazards which reduces the window of time to react to the hazard. This indicates that perception is the most individually vulnerable vehicular component.
- For two vulnerability modes, the combination of ‘DelayScale’ and ‘HazardScale’ in the most significant, as the vehicle now not only has difficulty reacting to hazards on time, but the number of hazards not being perceived means that there are many more hazards being detected late or not at all which can increase the risk of accidents.

Combined attacks against perception elements pose a great safety risk as the hazards will not only fail to be detected but also delayed in their detection.

- For three vulnerability modes under attack, 'AccidentScale', 'DelayScale' and 'HazardScale' have the greatest safety implications. This is because now there is weaker hazard detection as well as late hazard detected coupled with the compromising of the safety monitor integrity through 'AccidentScale'. This means that the vehicle is not only dealing with hazards poorly, but the safety policies are unable to act to ensure safe driving throughout all this. Attacks on perception at the same time as the safety monitor significantly increases risk as the vehicle is not only less capable of 'seeing' the surroundings but cannot even enforce its safety policy properly.
- For the four vulnerability modes, it is the combination of 'AccidentScale', 'HazardScale', 'DelayScale' and 'LateCH2AccScale' that causes the greatest risk to safety, as this brings the effects of the greatest risk for three modes and adds a new element of 'LateCH2AccScale' meaning that overlooked hazards may be less likely to be detected by the system later, essentially making most of the critical systems unable to do their jobs properly. Increasing the number of compromised elements makes things worse, as expected.
- It was found that 'OH2CH_scale' has little effect towards safety, causing the combinations that include it to be less dangerous (e.g. in the case of 'DelayScale' + 'HazardScale', where this combination had more risk than compromising all modes). This is due to it only existing within the 'sojournTime' timed activity (i.e. the area dealing with later hazards) at the same time as 'DelayScale' but primarily as a denominator of the probability calculations in the code which can cause it to lower the probability of success in different branches. The attack combinations are not all necessarily valid as there cannot be combinations that offer greater risk than compromising the entire system, as attacking more elements should consistently make things worse.
- Generally, activating all modes can still be considered a realistic worst case, but it seems that the modes on their own do not fully reflect a real CAV so an architecture-based model of similar design should be used in the next stages.
- As a 'realistic' model, it has much lower values in terms of risk of an accident, but this is attributed to the extremely low probability of successful attacks in the attack model. The parameters are based on attacks on computers rather than CAV so this should be considered when considering its application to other models.
- There is only a single absorbing state, so the results are limited to one type of accident, when there could be many different scenarios where a safety failure is not necessarily catastrophic, limiting the extent to which the model can assess safety.
- A limitation of the model is that it assumes adversaries that occur at a fixed rate with some success probability, which is typically not 100% meaning that the model does not account for an adversary that can perfectly execute an attack every time without issue (i.e. I assume a limited level of expertise from the adversary).
- It must also be noted that while the model is limited to Exponentially distributed timed activities only, this is not a major issue as according to the findings of (Littlewood, 1979), all Markov models of this type are asymptotically Exponential

irrespective of the choice of distribution for timed activities meaning that my results for the risk would give an Exponential result over time anyway.

4.5.3 Sensitivity Analysis

The parameters in the experiment section were either calculated or obtained through existing papers describing the rates of vehicle behaviours, or, when this was not available, through sensitivity analysis. This is where the unknown parameter value is varied to observe how it affects the overall result of the system, in this case CAV. To avoid an excess of results, this was varied when all vulnerability modes were active at once, so not all the experimented values were demonstrated. This uses the ITSC method (Popov *et al.*, 2022) as a quantitative means of determining suitable parameters as safety-security analysis must deal with uncertainty in both the safety and attack related parameters.

Parameter	Value(s)
AccidentScale	[1.0, 10.0]
CH2AccRate	1.0E-4
CH2OKRate	856.3
DelayScale	[1.0, 1000.0]
FH2AccRate	1.0E-5
FH2OKRate	856.3
HazardRate	197.4
HazardScale	[1.0, 10.0]
LateCH2AccScale	[1.0, 10.0]
MH2AccRate	0.001
MH2CHRate	2250.0
OH2CH scale	[1.0, 10.0]
attackRate	[0.1, 0.2, 0.3]
attackReset	[0.01, 1.0, 2.0]
attackSuccess	[0.05, 0.1, 0.2]
falseHazardProb	[0.0, 1.0E-4, 0.001]
isRSSperfect	0
missHazardProb	0.05

Table 19: Sensitivity Analysis Parameters – Vulnerability Modes

As seen in Table 19, the parameters varied were initially the vulnerability modes, but the ones that were unknown at this stage were the rate of attack, rate of reset, probability of attack success and the probability of a false hazard occurring, just to see if false hazards have any effect.

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
FHProb=0.0	0.2486	0.9793	0.9994	1.0000	1	1	1	1	1	1
FHProb=0.001	0.2484	0.9792	0.9994	1.0000	1	1	1	1	1	1
FHProb=1.0E-4	0.2486	0.9793	0.9994	1.0000	1	1	1	1	1	1

Table 20: False Hazard Sensitivity Data (All Modes)

Looking at these results, Table 20 shows very little change in the presence of a false hazard. This means that they have little effect on the model and the probability could be kept at a value of 0, but this assumes that false alarms are non-existent, which is simply not true, so the probability of false hazards was be set at 1.0E-4, meaning at least one false alarm every 10000 hours.

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
AttackRate=0.1	0.2486	0.9793	0.9994	1.0000	1	1	1	1	1	1
AttackRate=0.2	0.2761	0.9801	0.9995	1.0000	1	1	1	1	1	1
AttackRate=0.3	0.2850	0.9804	0.9995	1.0000	1	1	1	1	1	1

Table 21: Attack Rate Sensitivity Data (All Modes)

For the attack rates, Table 21 indicates that as expected more attacks does mean increased risk to safety as there are more attacks that may succeed for the same probability of success, so depending on the strength of adversary desired, this can be set to a high or low value. In this case, the middle value of 0.2 is chosen, so that at least two attacks occur every 10 hours.

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
AttackReset=0.01	0.2192	0.9600	0.9980	0.9999	1.0000	1	1	1	1	1
AttackReset=1.0	0.0187	0.1891	0.3299	0.4463	0.5424	0.6219	0.6875	0.7418	0.7866	0.8237
AttackReset=2.0	0.0107	0.1120	0.2029	0.2846	0.3578	0.4236	0.4826	0.5356	0.5831	0.6258

Table 22: Attack Reset Sensitivity Data (All Modes)

For attack resets (Table 22), which are essentially a means of removing the effects of the adversary periodically as it is assumed that it would not last forever, faster cleansing of the adversary suggests a lower risk to safety as the system is compromised for less time.

Therefore, choosing from these parameters, the value of 2.0 was chosen meaning that an attack will be reset every half hour, so this is how long an adversary must affect the system before the attack activity reloads.

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
AttackSuccess=0.05	0.0693	0.9392	0.9978	0.9999	1.0000	1	1	1	1	1
AttackSuccess=0.1	0.1177	0.9690	0.9991	1.0000	1.0000	1	1	1	1	1
AttackSuccess=0.2	0.1792	0.9762	0.9993	1.0000	1.0000	1	1	1	1	1

Table 23: Attack Success Sensitivity Data (All Modes)

In Table 23, as expected increased attack success results in more actual effects from attacks. A weak adversary would have little to no success and in this model the chosen success probability is 0.2 to be able to have larger numbers to observe across different experiments.

4.6 Partitioned Model

Now, the model must be partitioned to reflect each subsystem rather than a mode to be closer to CAV architecture, as the internal components of CAVs consist of ECUs rather than a bunch of timed activities and vulnerability modes. Therefore, the model was partitioned to reflect the below Figure 13. This step is necessary especially as a preparation for introducing defences at the level of ECU components. The models were simulated across multiple points in time from 100-9100 hours in 1000-hour steps, using a numeric solver with a 10^{-9} precision, as the risk of an accident evolves over time when observing CAVs. The full model documentation can be found in Appendix 13.3.

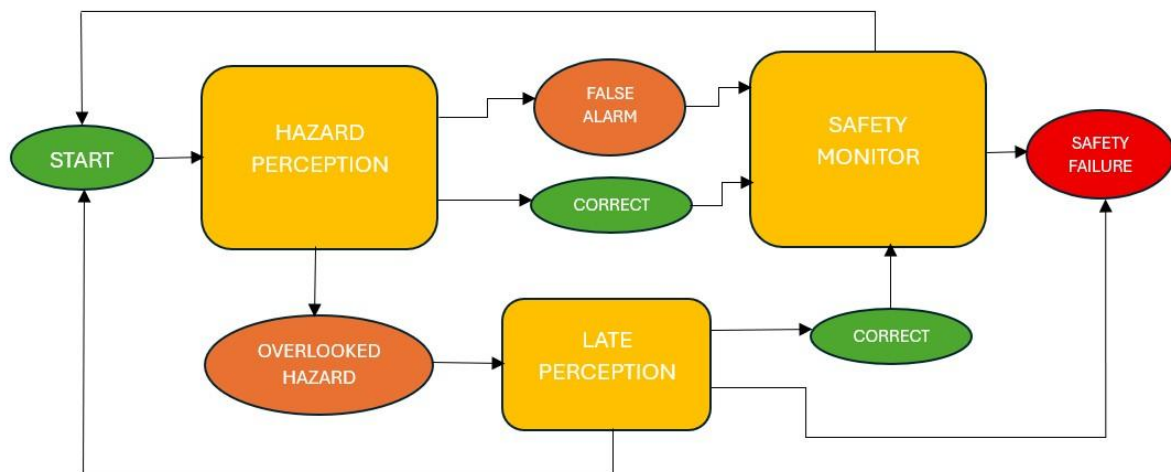


Figure 13: CAV Partition Block Diagram

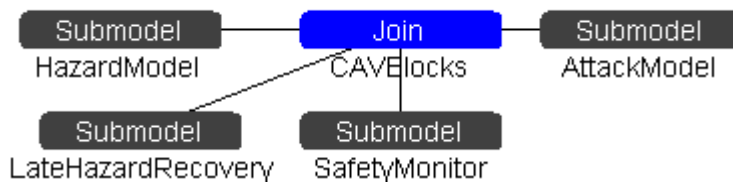


Figure 14: Composed Model with Partitions

In this case, the partitions are joined up to represent the same previous structure of the model, but each now only containing its own vulnerability modes, as seen in Figure 14. Therefore, whenever a partition is activated, all vulnerability modes in that section are active as the partition is now being attacked. The attack model remains the same at this stage, so this does not need to be described again.

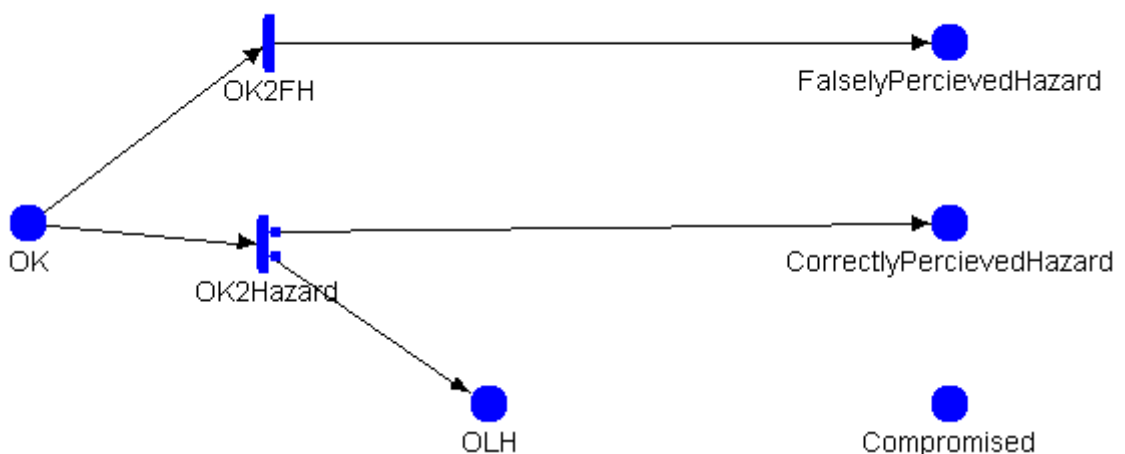


Figure 15: Hazard Partition

The first section (Figure 15) is that hazard perception component of the module, where hazards will either be false or real, which will either lead to falsely perceived hazards, correctly perceived hazards, or if the system fails to see it, an overlooked hazard.

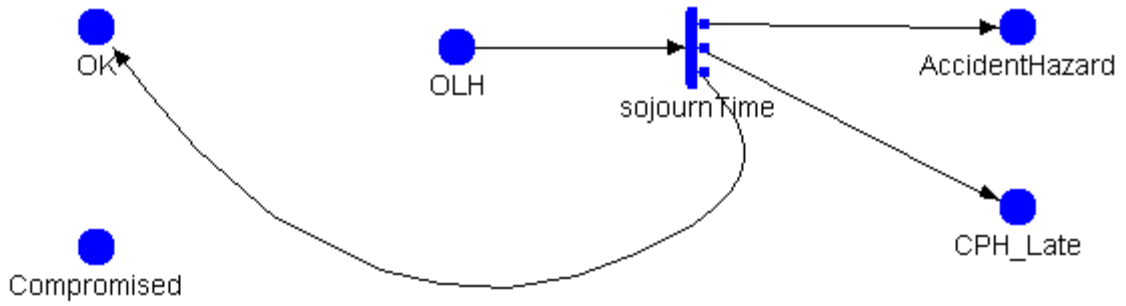
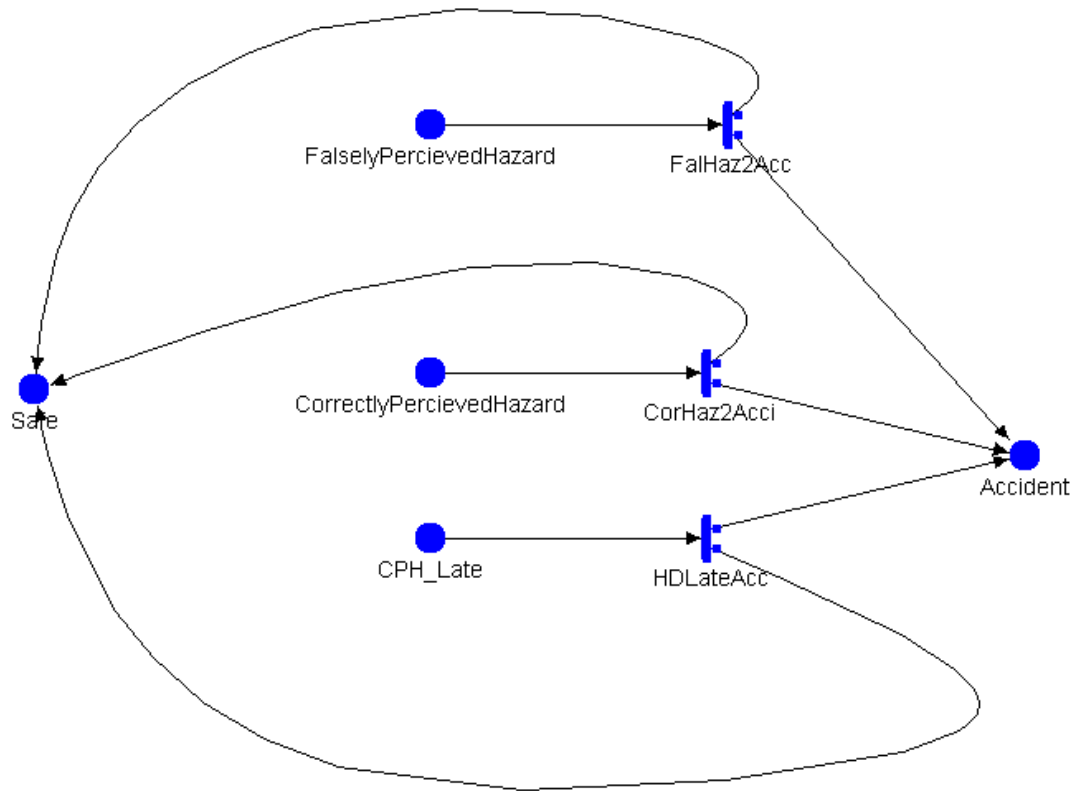


Figure 16: Overlooked Hazard Partition

This next component is for overlooked hazards (Figure 16), where an overlooked hazard will either lead to an accident or be perceived correctly later. If this hazard goes away earlier than when the vehicle needs to react, the system will go back to being OK.



isRSSperfect - a boolean variable:

1: if RSS is perfect (detected hazard cannot lead to accident)

0: if RSS is not perfect, then the hazard can lead to a an accident.

Compromised

Figure 17: Safety Monitor Partition

Finally, the safety monitor (Figure 17) where the RSS policy is usually implemented is present to check if the vehicle is obeying safety rules properly. If this partition is compromised, it should make it harder for the vehicle to check itself for safety.

By having these partitions, the admittedly abstract model is still grouped into elements that can represent ECUs or domains, meaning that it is like domain-based architecture. While it does not reflect all potential architectures and elements, it is sufficiently abstract where it does not lean too much into any specific architecture meaning that it can be used to model a generic in-vehicle network.

Attacks that attack partitions can be considered as attacks that compromise the CAN bus to reach ECUs and can then successfully compromise ECUs which could have different effects on safety, which was later used to more effectively model layered approaches to defence (e.g. defence-in-depth).

The partitions will be toggled using the newly introduced parameters that act as flags (enableHazard, enableOLH, enableSM), which allows each partition to be tested separately and in combination to see the effects with and without attacks. This also groups relevant modes into their own relevant partition, for which parameters are seen in the table below as the equivalent of how it would look if toggling modes separately or in combination.

Partition	AccidentScale	DelayScale	HazardScale	LateCH2AccScale	OH2CH_scale
Hazard	1.0	1.0	10.0	1.0	1.0
Overlooked	1.0	1000.0	1.0	1.0	10.0
SafetyMonitor	10.0	1.0	1.0	10.0	1.0
HazardOLH	1.0	1000.0	10.0	1.0	1.0
HazardSM	10.0	1.0	10.0	10.0	1.0
OverlookedSM	10.0	1000.0	1.0	10.0	10.0
All	10.0	1000.0	10.0	10.0	10.0
Baseline	1.0	1.0	1.0	1.0	1.0

Table 24: Partition Vulnerability Mode Equivalents

4.6.1 Main Model

Now, the model must be partitioned to reflect each subsystem rather than a mode to be more realistic in terms of CAV architecture, as the internal components of CAVs consist of ECUs rather than a bunch of timed activities and modes. These will be toggled using the newly introduced parameters that act as flags (enableHazard, enableOLH, enableSM), which allows each partition to be tested separately and in combination to see the effects. This also groups relevant modes into their own relevant partition, for which parameters are seen in the table below.

Parameter	Value(s)
AccidentScale	10.0
CH2AccRate	1.0E-4
CH2OKRate	856.3
DelayScale	1000.0
FH2AccRate	1.0E-5
FH2OKRate	856.3
HazardRate	197.4
HazardScale	10.0
LateCH2AccScale	10.0
MH2AccRate	0.001
MH2CHRate	2250.0
OH2OKRate	1000.0
OH2CH scale	10.0
attackRate	0.1
attackReset	2.0
attackSuccess	0.2
enableHazard	[0, 1]
enableOLH	[0, 1]
enableSM	[0, 1]
falseHazardProb	1.0E-4
isRSSperfect	0
missHazardProb	0.05

Table 25: Model Parameters of the System with Partitions

The values in Table 25 are used as the rates of events and probabilities in the timed activities that make up the SAN model. Apart from the attack parameters the rest were obtained from (Popov et al., 2022). The vulnerability modes were assigned in a way to increase the probability of a safety failure by at least one order of magnitude, while the attacks were determined through sensitivity analysis.

Based off these new flag parameters as opposed to vulnerability modes, each partition can be disabled to remove the effect of an attack on it, which means there are now eight experiments to observe, as seen in the table below.

Experiment	enableHazard	enableOLH	enableSM
Baseline	0	0	0
Hazard	1	0	0
Overlooked	0	1	0
SafetyMonitor	0	0	1
HazardOLH	1	1	0
HazardSM	1	0	1
OverlookedSM	0	1	1
AllParts	1	1	1

Table 26: List of Partition Experiments

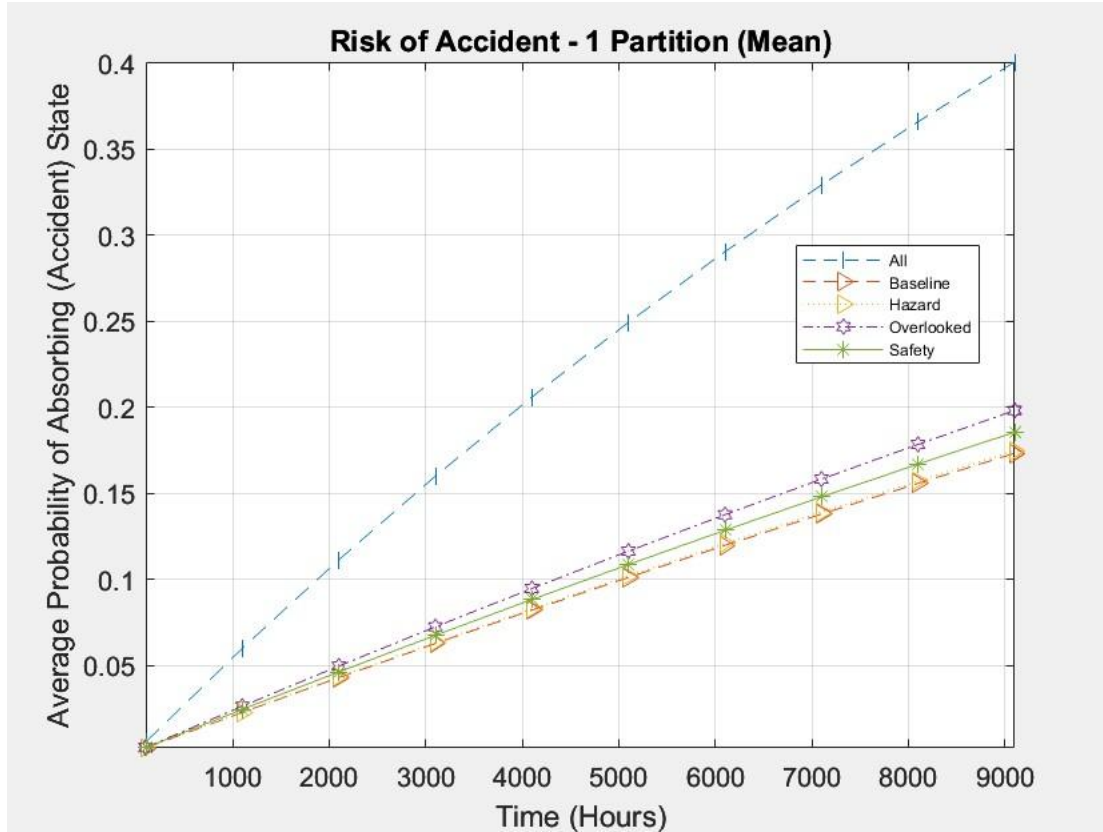


Figure 18: Single Partition Attacks

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
Hazard	0.0021	0.0230	0.0434	0.0634	0.0829	0.1021	0.1209	0.1392	0.1572	0.1748
Overlooked	0.0024	0.0264	0.0497	0.0725	0.0947	0.1165	0.1377	0.1583	0.1785	0.1982
SafetyMonitor	0.0023	0.0245	0.0463	0.0676	0.0884	0.1087	0.1286	0.1480	0.1670	0.1856
All	0.0056	0.0600	0.1114	0.1601	0.2060	0.2495	0.2905	0.3294	0.3661	0.4008

Table 27: Individual Compromised Partition Data

For individual partitions shown in Figure 18 and Table 27, the highest to lowest risk partitions in terms of safety are as follows:

1. Overlooked (Equivalent of 'DelayScale' and 'OH2CH_scale' vulnerability modes combined): Deals with a late or overlooked hazard if possible.
2. Safety (Combination of 'AccidentScale' and 'LateCH2AccScale' vulnerability modes): Monitors safety to limit the probability of going to an accident.
3. Hazard (Contains 'HazardScale' parameter): Used for perception of real and false hazards.

This means that attacking the overlooked hazard perception component is the most dangerous alone, followed by compromising the safety monitor, as these would both be capable of compensating for any attack that interfered with the main hazard perception on its own. All these combinations remain below attacking all partitions in terms of risk, meaning that combination attacks must once again be explored.

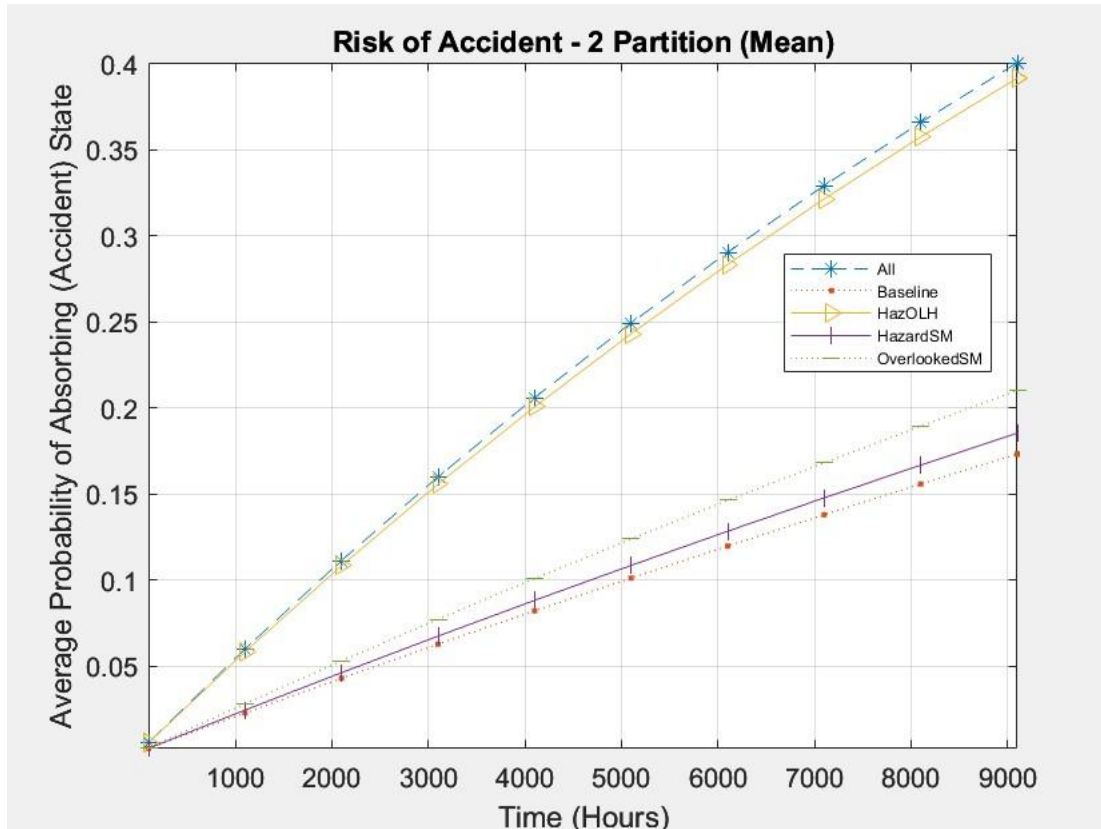


Figure 19: Combined Partitions Attacked

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
HazardOLH	0.0054	0.0583	0.1084	0.1558	0.2007	0.2432	0.2835	0.3216	0.3576	0.3918
HazardSM	0.0023	0.0245	0.0462	0.0675	0.0883	0.1086	0.1284	0.1479	0.1668	0.1854
OverlookedSM	0.0026	0.0281	0.0530	0.0773	0.1009	0.1239	0.1464	0.1682	0.1895	0.2103
All	0.0056	0.0600	0.1114	0.1601	0.2060	0.2495	0.2905	0.3294	0.3661	0.4008

Table 28: Two Compromised Partitions Data

Results from Table 28 and Figure 19 show that combination attacks have increased risk, which are ranked from lowest to highest as follows:

1. HazOLH (Hazard Perception + Overlooked Hazards)
2. OverlookedSM (Overlooked Hazard + Safety Monitor)
3. HazardSM (Hazard Perception + Safety Monitor)

This was expected as it followed on from the individual partitions where ‘Overlooked’ (or OLH) was the most dangerous, and now that it is combined with the ‘Hazard’ perception it has become even more risky, only slightly below the compromising of all partitions at once, as this essentially means all hazard perception elements are at risk meaning that hazards will be far more difficult to deal with, leaving everything to the safety monitor.

When compromising in combination, the most significant combination is to compromise all parts of the vehicle. However, as a pair, compromising the hazard perception and late hazard perception is the most notable area in which safety implications occur due to an attack. Therefore, defence should either protect all components, or if this is not an option, protect anything involving hazard perception.

4.6.1.1 Discussion

The results from attacking partitions indicate the following:

- Attacks on the late/overlooked hazard perception component of the vehicle have the most individual effect. This is because the vehicle will miss its fair share of hazards on time, especially in a constantly changing environment so if the unit designed to compensate for this is not working properly then the risk to safety will be very high. Therefore, this suggests that attacks against a perception module are the most individually dangerous, as they can cause a hazard not to be seen on time, or in some cases, not at all.
- Compromising the hazard perception with the late/overlooked hazard system at the same time is the most dangerous combination of two components being compromised. This is because the detection rate of hazards deteriorates while the ones not immediately detected are also not being detected as often, so the perception is heavily weakened meaning that the CAV is liable to have a lot more accidents. Combined attacks against both perception modules are the most dangerous pair of attacks indicating that the perception modules are where the strongest defences should be applied.
- Compromising all components of the vehicle together is still the worst-case scenario here, as doing this means that not only is the perception not working properly, but the safety monitor cannot attempt to enforce safety for the vehicle either.
- There is only a single absorbing state, so the results are limited to one type of accident, when there could be many different scenarios where a safety failure is not necessarily catastrophic, limiting the extent to which the model can assess safety.
- A limitation of the model is that it assumes adversaries that occur at a fixed rate with some success probability, which is typically not 100% meaning that the model does not account for an adversary that can perfectly execute an attack every time without issue (i.e. I assume a limited level of expertise from the adversary).
- It must also be noted that while the model is limited to Exponentially distributed timed activities only, this is not a major issue as according to the findings of (Littlewood, 1979), all Markov models of this type are asymptotically Exponential regardless of the distributions used meaning that my results for the risk would give an Exponential result over time anyway.

4.6.2 Realistic Attack

Continuing from the previously used realistic attacks, they have now been applied to the partitioned model as well, using the parameters from the Table 14 below. The additional graphs obtained from the simulations can be found in Appendix 14.2.

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
Hazard	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1199	0.1381	0.1559	0.1734
Overlooked	0.0021	0.0229	0.0433	0.0632	0.0827	0.1018	0.1205	0.1389	0.1568	0.1744
SafetyMonitor	0.0021	0.0228	0.0431	0.0630	0.0824	0.1015	0.1202	0.1384	0.1563	0.1738
All	0.0022	0.0243	0.0458	0.0669	0.0875	0.1076	0.1274	0.1466	0.1655	0.1839

Table 29: Raw Data - Realistic Attack (Single Partitioned)

For these parameters, the results in Table 29 indicate that there is very little difference when it comes to compromising individual partitions, with ‘Overlooked’ hazard perception having the highest effect, but remaining very close to baseline much like the others and being below

the risk from compromising all components together, meaning that combinations are the next area to explore.

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
HazardOLH	0.0022	0.0242	0.0457	0.0667	0.0873	0.1074	0.1270	0.1462	0.1650	0.1834
HazardSM	0.0021	0.0228	0.0431	0.0630	0.0824	0.1015	0.1202	0.1384	0.1563	0.1738
OverlookedSM	0.0021	0.0230	0.0434	0.0634	0.0830	0.1021	0.1209	0.1393	0.1572	0.1749
All	0.0022	0.0243	0.0458	0.0669	0.0875	0.1076	0.1274	0.1466	0.1655	0.1839

Table 30: Raw Data - Realistic Attack (Combined Partitions)

Table 30 shows that the combination of hazard perception and overlooking hazards (HazOLH) has the most serious effect, almost at the same level as compromising all three partitions at once, implying that the safety monitor has a less serious role to play than the perception itself, which means this is where defences should be deployed to ensure fewer safety implications due to security problems.

4.6.2.1 Discussion

The results from this section indicate the following:

- Attacks on the late/overlooked hazard perception component of the vehicle have the most individual effect. This is because the vehicle will miss its fair share of hazards on time, especially in a constantly changing environment so if the unit designed to compensate for this is not working properly then the risk to safety will be very high. Therefore, perception is a very at-risk component in CAVs where defences should be applied.
- Compromising the hazard perception with the late/overlooked hazard system at the same time is the most dangerous combination of two components being compromised. This is because the detection rate of hazards deteriorates while the ones not immediately detected are also not being detected as often, so the perception is heavily weakened meaning that the CAV is liable to have a lot more accidents. This extends the previous point, indicating that any component involved in perception and dealing with hazards directly should be defended more than the others.
- Compromising all components of the vehicle together is still the worst-case scenario here, as doing this means that not only is the perception not working properly, but the safety monitor cannot attempt to enforce safety for the vehicle either.
- There is only a single absorbing state, so the results are limited to one type of accident, when there could be many different scenarios where a safety failure is not necessarily catastrophic, limiting the extent to which the model can assess safety.
- A limitation of the model is that it assumes adversaries that occur at a fixed rate with some success probability, which is typically not 100% meaning that the model does not account for an adversary that can perfectly execute an attack every time without issue (i.e. I assume a limited level of expertise from the adversary).
- It must also be noted that while the model is limited to Exponentially distributed timed activities only, this is not a major issue as according to the findings of (Littlewood, 1979), all Markov models of this type are asymptotically Exponential regardless of my choice of distribution meaning that my results for the risk would give an Exponential result over time anyway.

4.6.3 Sensitivity Analysis

At this stage, some more realistic parameters were obtained, except for the rate at which attacks were reset. Therefore, the parameter was set to be varied as shown in the table below. This analysis increases the reset rate by orders of magnitude at first and then proceeds to double it once it is within the range of an hour.

Parameter	Value(s)
AccidentScale	10.0
CH2AccRate	1.0E-4
CH2OKRate	856.3
DelayScale	1000.0
FH2AccRate	1.0E-5
FH2OKRate	856.3
HazardRate	197.4
HazardScale	10.0
LateCH2AccScale	10.0
MH2AccRate	0.001
MH2CHRate	2250.0
OH2OKRate	1000.0
OH2CH scale	10.0
attackRate	0.1
attackReset	[0.001,0.01,0.1,1.0,2.0,4.0]
attackSuccess	0.2
enableHazard	[0, 1]
enableOLH	[0, 1]
enableSM	[0, 1]
falseHazardProb	1.0E-4
isRSSperfect	0
missHazardProb	0.05

Table 31: Sensitivity Analysis Parameters – Partitioned Model

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
Reset=0.001	0.1758	0.9716	0.9990	1.0000	1.0000	1	1	1	1	1
Reset=0.01	0.1469	0.9148	0.9915	0.9992	0.9999	1.0000	1.0000	1	1	1
Reset=0.1	0.0540	0.4822	0.7165	0.8448	0.9151	0.9535	0.9746	0.9861	0.9924	0.9958
Reset=1.0	0.0090	0.0950	0.1736	0.2453	0.3108	0.3707	0.4253	0.4752	0.5208	0.5624
Reset=2.0	0.0056	0.0600	0.1114	0.1601	0.2060	0.2495	0.2905	0.3294	0.3661	0.4008
Reset=4.0	0.0039	0.0417	0.0781	0.1131	0.1467	0.1791	0.2103	0.2403	0.2691	0.2968

Table 32: Attack Reset Sensitivity - Raw Data (All Partitions)

Once again, as expected, the higher attack reset is better for safety as there is less time to affect the system, so leaving it at around 2.0 is sufficient for a short window, but this could also be set to 4.0 for shorter attacks.

5 Uncertainty in Model Parameters

So far, all the distributions used in timed activities in the SAN models have been using the Exponential distribution. While this is convenient as it allows one to solve the models using the numeric solvers available in Mobius, assuming all timed activities included in the model(s) exponentially distributed may be questionable. For instance, if all attacks will follow a Poisson process (with exponentially distributed intervals between the attacks) may seem unjustifiable.

In this chapter I studied the implications of assuming that the attacks will deviate from a Poisson process and the intervals between attacks may follow a distribution different from exponential. After a discussion with supervisors, it was decided that a plausible model would be a model of attacks whereby the intervals between attacks will be exponential distributions with uncertain parameter, i.e. the parameter λ of an exponential distribution will be treated as a random variable with its own distribution such as Gamma. The resulting intervals between the events (attacks) of such compound distribution would be the Lomax distribution. Since this distribution is not among the distributions provided by Mobius/SAN the Lomax distribution had to be implemented. This was done by changing the respective timed activities in the affected models from “Exponential” to ‘Deterministic’ and implementing a Lomax distribution, a Type 2 Pareto which starts support at the value of 0. The models were simulated across multiple points in time from 100-9100 hours in 1000-hour steps, using Monte Carlo simulations for 100 million batches in steps of 1000 batches to observe the change in accident probability over the period for which the CAV is observed. The full model documentation can be found in Appendix 13.5.

5.1 The Lomax Distribution

The Lomax distribution is a Type 2 Pareto distribution which starts support at the value of 0. For the rate parameter (λ) this is the formula now used in the timed activities when they are set to ‘Deterministic’ with a Lomax distribution:

$$(1) x_m = \frac{\alpha-1}{\lambda}$$

$$(2) \text{Lomax} = \text{Pareto}(x_m, \alpha) - x_m$$

For the equations (1) and (2), this is how it is now set up and coded wherever the Lomax is set up, where α is a parameter that is greater than 1 and may be varied to see the effects in this case. This model is equivalent to the exponential distribution in terms of expected value and accounts for any uncertainty that may exist in the attack model as it is difficult to truly know the behaviour of the adversary in all cases.

For this part of the model, the value of α was varied between 2, 5 and 10. In all these cases it is shown how the risk of an accident is varied compared to a purely Exponential model for different attacks on partitions while accounting for the uncertainty in the attack model. The table below shows the chosen parameters, where the value of α is only applied to the ‘Ready2Attack’ activity which models the intensity and success of an adversary.

Parameter	Value(s)
AccidentScale	10.0
CH2AccRate	1.0E-4
CH2OKRate	856.3
DelayScale	1000.0
FH2AccRate	1.0E-5
FH2OKRate	856.3
HazardRate	197.4
HazardScale	10.0
LateCH2AccScale	10.0
MH2AccRate	0.001
MH2CHRate	2250.0
OH2OKRate	1000.0
OH2CH scale	10.0
Alpha	[2.0,5.0,10.0]
attackRate	0.1
attackReset	2.0
attackSuccess	0.2
enableHazard	[0, 1]
enableOLH	[0, 1]
enableSM	[0, 1]
falseHazardProb	1.0E-4
isRSSperfect	0
missHazardProb	0.05

Table 33: Lomax Attack Parameters

Once the Lomax was implemented, I repeated the experiments developed early to estimate the impact of the uncertainty added to the rate attacks on system safety. To make the comparison fair a decision was taken that the mean value of the intervals between attacks will not be affected by the change of the distribution of the inter-attack time from Exponential to Lomax, e.g. $E_{Exp}[X_m] = E_{Lomax}[X_m]$.

The results are shown below. Since the probability of accident is computed via Monte Carlo simulations, all plots show the probability of accident together with the respective 95% confidence level.

5.1.1 Attack Models

The attack model in this case has been set to Deterministic, sampling from a composite Pareto in the case of Lomax, which sampling Exponential where I wish to compare to this distribution. This is due to how the model was set up, where the sampled and non-sampled versions of the distributions gave different results, so by running both Deterministic and with Monte Carlo simulation it was possible to avoid an unnecessary difference and compare them properly. These were both used for the comparison only and are seen in the tables below.

Timed Activity:	Ready2Attack
Distribution Parameters	Value static UserDistributions dist(0,31415); double lambda = attackRate; double delay; double xm; xm = (alpha - 1) / lambda; delay = dist.Pareto(xm,alpha) - xm; return(delay);
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 return(attackSuccess); case 2 return(1.0 - attackSuccess);

Table 34: Lomax Distribution Attack Model

Timed Activity:	Ready2Attack
Distribution Parameters	Value static UserDistributions dist(0,31415); double lambda = attackRate; double delay; delay = dist.Exponential(lambda); return(delay);
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 return(attackSuccess); case 2 return(1.0 - attackSuccess);

Table 35: Exponential Distribution Attack Model

5.2 Lomax vs. Exponential

Here, the different values of α for the Lomax are compared to the previous Exponentially distributed SAN model of the same structure, so that it can be seen how much of a difference exists between the Lomax and Exponential distributions to see the impact of the uncertainty of the attack model in terms of safety. As this used Monte Carlo, two tables were presented below each figure, with the first being the risk of an accident (raw data) and the second immediately being the confidence level.

5.2.1 Results

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0132	0.0335	0.0534	0.0733	0.0920	0.1102	0.1284	0.1471	0.1643	0.1808
$\alpha=2.0$	0.0131	0.0335	0.0534	0.0735	0.0923	0.1105	0.1290	0.1473	0.1643	0.1812
$\alpha=5.0$	0.0136	0.0339	0.0532	0.0728	0.0916	0.1098	0.1285	0.1466	0.1638	0.1805
$\alpha=10.0$	0.0136	0.0340	0.0537	0.0733	0.0922	0.1100	0.1286	0.1469	0.1644	0.1812

Table 36: Lomax vs. Exponential - Raw Data (Baseline)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	6.6007e-04	0.0010	0.0013	0.0015	0.0017	0.0018	0.0019	0.0020	0.0021	0.0022
$\alpha=2.0$	6.5378e-04	0.0010	0.0013	0.0015	0.0017	0.0018	0.0019	0.0020	0.0021	0.0022
$\alpha=5.0$	6.7429e-04	0.0011	0.0013	0.0015	0.0017	0.0018	0.0020	0.0021	0.0022	0.0022
$\alpha=10.0$	6.7719e-04	0.0011	0.0013	0.0015	0.0017	0.0018	0.0020	0.0021	0.0022	0.0023

Table 37: Lomax vs. Exponential - Raw Data - Confidence Levels (Baseline)

Looking at the baseline comparison (Table 36, Table 37) it is seen that the values are quite close with overlapping error bars as expected, but can be listed from highest to lowest as follows:

1. $\alpha=10.0$
2. $\alpha=2.0$
3. Baseline (Exponential)
4. $\alpha=5.0$

The baselines are the same, with only small differences in some cases of the Lomax which suggest that the uncertainty is negligible in the absence of attacks.

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Hazard	0.0131	0.0336	0.0537	0.0735	0.0926	0.1110	0.1296	0.1482	0.1655	0.1825
$\alpha=2.0$	0.0126	0.0335	0.0531	0.0733	0.0922	0.1106	0.1296	0.1485	0.1659	0.1827
$\alpha=5.0$	0.0132	0.0338	0.0536	0.0733	0.0926	0.1112	0.1302	0.1485	0.1660	0.1827
$\alpha=10.0$	0.0134	0.0338	0.0541	0.0739	0.0930	0.1113	0.1301	0.1483	0.1655	0.1823

Table 38: Lomax vs. Exponential - Raw Data (Hazard Perception)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Hazard	6.5244e-04	0.0010	0.0013	0.0015	0.0017	0.0018	0.0019	0.0020	0.0021	0.0022
$\alpha=2.0$	6.2796e-04	0.0010	0.0013	0.0015	0.0016	0.0018	0.0019	0.0020	0.0021	0.0022
$\alpha=5.0$	6.6029e-04	0.0010	0.0013	0.0015	0.0017	0.0018	0.0019	0.0021	0.0022	0.0022
$\alpha=10.0$	6.6669e-04	0.0010	0.0013	0.0015	0.0017	0.0018	0.0020	0.0021	0.0022	0.0022

Table 39: Lomax vs. Exponential - Raw Data - Confidence Intervals (Hazard Perception)

Looking at the hazard perception (Table 38, Table 39) it is seen that the values are quite close with overlapping error bars as expected, but can be listed from highest to lowest as follows:

1. $\alpha=5.0$
2. $\alpha=2.0$
3. Hazard (Exponential)
4. $\alpha=10.0$

The hazard perception values for Exponential and Lomax are very close here. The uncertainty depending on the Lomax parameters will give a negligible increase or decrease in comparison that has no serious effect on the result.

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Overlooked	0.0135	0.0370	0.0600	0.0825	0.1036	0.1246	0.1454	0.1663	0.1858	0.2050
$\alpha=2.0$	0.0135	0.0370	0.0593	0.0821	0.1035	0.1246	0.1459	0.1664	0.1859	0.2051
$\alpha=5.0$	0.0135	0.0373	0.0608	0.0830	0.1044	0.1254	0.1462	0.1665	0.1862	0.2051
$\alpha=10.0$	0.0139	0.0376	0.0601	0.0833	0.1047	0.1256	0.1466	0.1675	0.1868	0.2061

Table 40: Lomax vs. Exponential - Raw Data (Overlooked Hazard)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Overlooked	6.7675e-04	0.0011	0.0014	0.0016	0.0018	0.0019	0.0021	0.0022	0.0023	0.0024
$\alpha=2.0$	6.7697e-04	0.0011	0.0014	0.0016	0.0018	0.0019	0.0021	0.0022	0.0023	0.0024
$\alpha=5.0$	6.7211e-04	0.0011	0.0014	0.0016	0.0018	0.0019	0.0021	0.0022	0.0023	0.0024
$\alpha=10.0$	6.9210e-04	0.0011	0.0014	0.0016	0.0018	0.0020	0.0021	0.0022	0.0023	0.0024

Table 41: Lomax vs. Exponential - Raw Data - Confidence Levels (Overlooked Hazard)

Looking at the overlooked hazards (Table 40, Table 41) it is seen that the values are quite close with overlapping error bars as expected, but can be listed from highest to lowest as follows:

1. $\alpha=10.0$
2. $\alpha=5.0$
3. $\alpha=2.0$
4. Overlooked (Exponential)

The values are once again close here, but the Exponential is the lowest in comparison to all the Lomax, meaning that on a very small level the Lomax uncertainty increases the risk.

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
SafetyMonitor	0.0134	0.0354	0.0566	0.0778	0.0983	0.1179	0.1380	0.1575	0.1759	0.1933
$\alpha=2.0$	0.0129	0.0348	0.0563	0.0772	0.0972	0.1166	0.1363	0.1557	0.1744	0.1920
$\alpha=5.0$	0.0138	0.0355	0.0573	0.0786	0.0981	0.1182	0.1379	0.1573	0.1756	0.1936
$\alpha=10.0$	0.0132	0.0352	0.0564	0.0780	0.0981	0.1174	0.1375	0.1567	0.1748	0.1927

Table 42: Lomax vs. Exponential - Raw Data (Safety Monitor)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
SafetyMonitor	6.6841e-04	0.0011	0.0013	0.0016	0.0017	0.0019	0.0020	0.0021	0.0022	0.0023
$\alpha=2.0$	6.4382e-04	0.0010	0.0013	0.0015	0.0017	0.0018	0.0020	0.0021	0.0022	0.0022
$\alpha=5.0$	6.8964e-04	0.0011	0.0014	0.0016	0.0018	0.0019	0.0020	0.0022	0.0022	0.0023
$\alpha=10.0$	6.5612e-04	0.0011	0.0013	0.0015	0.0017	0.0019	0.0020	0.0021	0.0022	0.0023

Table 43: Lomax vs. Exponential - Raw Data - Confidence Levels (Safety Monitor)

Looking at the safety monitor (Table 42, Table 43) it is seen that the values are quite close with overlapping error bars as expected, but can be listed from highest to lowest as follows:

1. $\alpha=5.0$
2. SafetyMonitor (Exponential)
3. $\alpha=10.0$
4. $\alpha=2.0$

Very negligible differences are seen, suggesting that the results of the two distributions are equal. A fair amount of the Lomax tests for this partition seem to give lower risk than Exponential, but Lomax can still give higher risk, though this is negligible.

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
HazardOLH	0.0138	0.0660	0.1157	0.1629	0.2063	0.2480	0.2872	0.3246	0.3602	0.3936
$\alpha=2.0$	0.0154	0.0684	0.1182	0.1646	0.2089	0.2511	0.2909	0.3281	0.3639	0.3975
$\alpha=5.0$	0.0149	0.0669	0.1155	0.1628	0.2061	0.2477	0.2868	0.3239	0.3603	0.3937
$\alpha=10.0$	0.0149	0.0663	0.1157	0.1631	0.2069	0.2472	0.2877	0.3258	0.3604	0.3937

Table 44: Lomax vs. Exponential - Raw Data (Hazard + Overlooked)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
HazardOLH	6.8964e-04	0.0015	0.0019	0.0022	0.0024	0.0026	0.0027	0.0028	0.0028	0.0029
$\alpha=2.0$	7.6741e-04	0.0016	0.0020	0.0023	0.0025	0.0027	0.0028	0.0029	0.0030	0.0030
$\alpha=5.0$	7.4357e-04	0.0015	0.0020	0.0023	0.0025	0.0026	0.0028	0.0029	0.0029	0.0030
$\alpha=10.0$	7.4381e-04	0.0015	0.0020	0.0023	0.0025	0.0026	0.0028	0.0029	0.0029	0.0030

Table 45: Lomax vs. Exponential - Raw Data - Confidence Levels (Hazard + Overlooked)

Looking at the hazard and overlooked combination (Table 44, Table 45) it is seen that the values are quite close with overlapping error bars as expected, but can be listed from highest to lowest as follows:

1. $\alpha=2.0$
2. $\alpha=10.0$
3. $\alpha=5.0$
4. HazardOLH (Exponential)

Again, the Lomax shows a slightly higher risk for this combination, with all of them being higher than Exponential, but on a negligible level which suggests that the distributions can be considered equal in terms of effect here, so the uncertainty introduced is not a major concern.

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
HazardSM	0.0128	0.0344	0.0558	0.0773	0.0977	0.1171	0.1370	0.1565	0.1751	0.1926
$\alpha=2.0$	0.0131	0.0350	0.0569	0.0783	0.0983	0.1180	0.1380	0.1570	0.1757	0.1934
$\alpha=5.0$	0.0133	0.0351	0.0564	0.0773	0.0974	0.1170	0.1369	0.1566	0.1751	0.1927
$\alpha=10.0$	0.0129	0.0344	0.0559	0.0770	0.0975	0.1174	0.1368	0.1561	0.1747	0.1925

Table 46: Lomax vs. Exponential - Raw Data (Hazard + Safety Monitor)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
HazardSM	6.3824e-04	0.0010	0.0013	0.0015	0.0017	0.0018	0.0020	0.0021	0.0022	0.0022
$\alpha=2.0$	6.5506e-04	0.0011	0.0013	0.0015	0.0017	0.0019	0.0020	0.0021	0.0022	0.0023
$\alpha=5.0$	6.6583e-04	0.0011	0.0013	0.0016	0.0017	0.0019	0.0020	0.0021	0.0022	0.0023
$\alpha=10.0$	6.4465e-04	0.0010	0.0013	0.0015	0.0017	0.0018	0.0020	0.0021	0.0022	0.0022

Table 47: Lomax vs. Exponential - Raw Data - Confidence Levels (Hazard + Safety Monitor)

Looking at the hazard and safety monitor combination (Table 46, Table 47) it is seen that the values are quite close with overlapping error bars as expected, but can be listed from highest to lowest as follows:

1. $\alpha=2.0$
2. $\alpha=5.0$
3. HazardSM (Exponential)
4. $\alpha=10.0$

Once again the differences are very negligible, with small differences introduced by the uncertainty of the Lomax, but the distributions can be considered pretty much equal, so it does not matter which one is used for modelling.

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
OverlookedSM	0.0132	0.0377	0.0622	0.0866	0.1095	0.1316	0.1541	0.1759	0.1969	0.2172
$\alpha=2.0$	0.0131	0.0384	0.0630	0.0873	0.1096	0.1317	0.1547	0.1763	0.1970	0.2175
$\alpha=5.0$	0.0140	0.0394	0.0639	0.0882	0.1107	0.1329	0.1551	0.1773	0.1985	0.2188
$\alpha=10.0$	0.0137	0.0385	0.0634	0.0876	0.1107	0.1328	0.1546	0.1760	0.1968	0.2176

Table 48: Lomax vs. Exponential - Raw Data (Overlooked + Safety Monitor)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
OverlookedSM	6.5986e-04	0.0011	0.0014	0.0016	0.0018	0.0020	0.0021	0.0022	0.0023	0.0024
$\alpha=2.0$	6.5548e-04	0.0011	0.0014	0.0016	0.0018	0.0019	0.0021	0.0022	0.0023	0.0024
$\alpha=5.0$	7.0187e-04	0.0012	0.0015	0.0017	0.0019	0.0020	0.0022	0.0023	0.0024	0.0025
$\alpha=10.0$	6.8303e-04	0.0011	0.0014	0.0017	0.0018	0.0020	0.0021	0.0022	0.0023	0.0024

Table 49: Lomax vs. Exponential - Raw Data - Confidence Levels (Overlooked + Safety Monitor)

Looking at the overlooked hazard and safety monitor combination (Table 48, Table 49) it is seen that the values are quite close with overlapping error bars as expected, but can be listed from highest to lowest as follows:

1. $\alpha=5.0$
2. $\alpha=10.0$
3. $\alpha=2.0$
4. OverlookedSM

In this combination, the Lomax gives very small and negligible increases in the safety risk compared to Exponential, but the distributions are still the same because of how small these differences are.

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
All	0.0147	0.0686	0.1200	0.1672	0.2130	0.2569	0.2962	0.3346	0.3712	0.4058
$\alpha=2.0$	0.0157	0.0702	0.1216	0.1689	0.2132	0.2553	0.2956	0.3346	0.3712	0.4058
$\alpha=5.0$	0.0149	0.0686	0.1194	0.1678	0.2138	0.2567	0.2976	0.3356	0.3718	0.4060
$\alpha=10.0$	0.0156	0.0691	0.1198	0.1681	0.2132	0.2557	0.2958	0.3344	0.3705	0.4047

Table 50: Lomax vs. Exponential - Raw Data (All Partitions)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
All	7.3103e-04	0.0015	0.0020	0.0023	0.0025	0.0027	0.0028	0.0029	0.0029	0.0030
$\alpha=2.0$	7.8285e-04	0.0016	0.0021	0.0024	0.0026	0.0027	0.0029	0.0030	0.0030	0.0031
$\alpha=5.0$	7.4260e-04	0.0016	0.0020	0.0023	0.0025	0.0027	0.0028	0.0029	0.0030	0.0030
$\alpha=10.0$	7.8007e-04	0.0016	0.0020	0.0024	0.0026	0.0027	0.0029	0.0030	0.0030	0.0031

Table 51: Lomax vs. Exponential - Raw Data - Confidence Levels (All Partitions)

Looking at all partitions compromised (Table 50, Table 51) it is seen that the values are quite close with overlapping error bars as expected, but can be listed from highest to lowest as follows:

1. $\alpha=5.0$
2. $\alpha=2.0$
3. All (Exponential)
4. $\alpha=10.0$

Even when compromising all partitions, the differences between the two distributions are negligible, with the uncertainty only introducing small variations in the risk of an accident. Therefore, these can be considered interchangeable, and the use of Exponential is more than sufficient. This matches (Littlewood, 1979) which states that the time to an absorbing state for a semi-Markov model is asymptotically Exponential where the rate of an accident (i.e. to the absorbing state) is much smaller than the rate of transition to the non-absorbing states, even being able to provide a formula for how this works.

5.2.1.1 Discussion

Comparing the two distributions has given the following conclusions:

- Very negligible differences between the Lomax and Exponential results for both attacks, meaning that either distribution could be used.
- The uncertainty in the model parameters is negligible for the attack model as seen by the very limited differences.
- The equivalence of the two distributions in this model supports the theory by (Littlewood, 1979) of a semi-Markov model where the relationship between the two distributions is asymptotic, where the probability of going to the absorbing state is much smaller than the probability of going to non-absorbing states.
- This small difference means that the expected values of both distributions should also be the same and this is to be checked as part of validation.

6 Defences

In this section, new defences were explored, unlike the existing ‘defence’ in the system which has been a single cleansing built into the reset activity for the attack model, as this can only go so far when it comes to being effective.

6.1 Defence-in-Depth

This type of defence works by having layers of defences to make an adversary must do more work to be able to affect the system. The previous partitioned model has been given an extra layer, requiring the system to be compromised to reach those partitions before attempting to compromise them. Only then will any attack be considered successful and have any effect on the system. This is analogous to accessing the CAN bus and then attacking the individual components, which is represented by attack success probabilities both at the system and partition level in the model. The models were simulated across multiple points in time from 100-9100 hours in 1000-hour steps, using a numeric solver with a 10^{-9} precision, as the risk of an accident evolves over time when observing CAVs. The full model documentation can be found in Appendix 13.4.

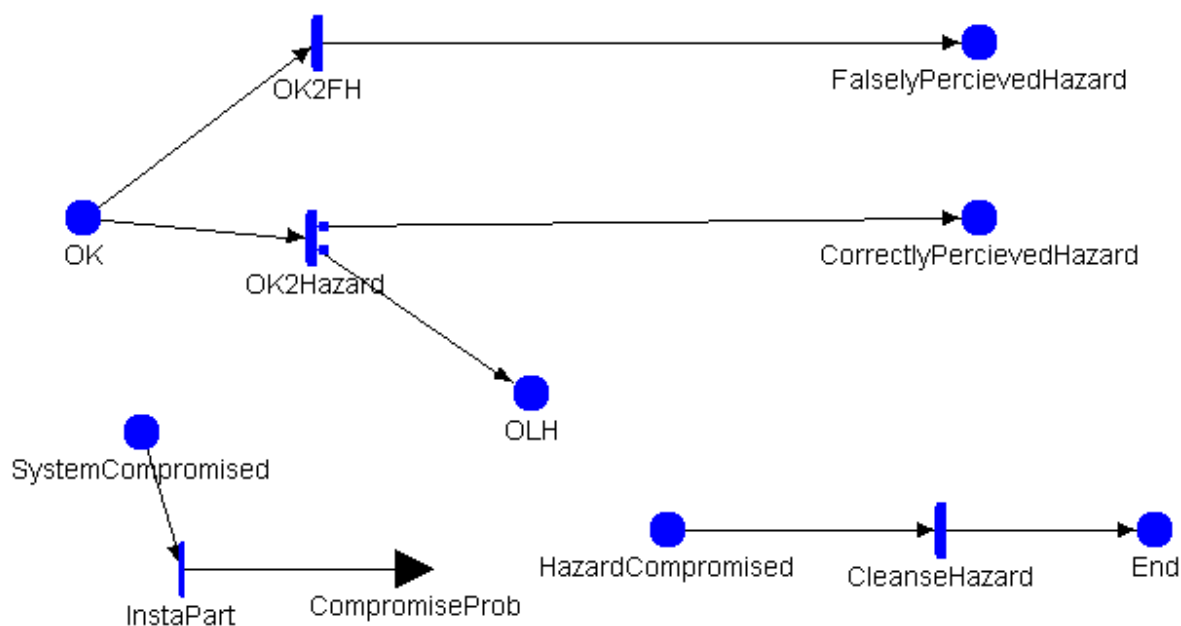


Figure 20: Hazard Perception - Defence-in-Depth

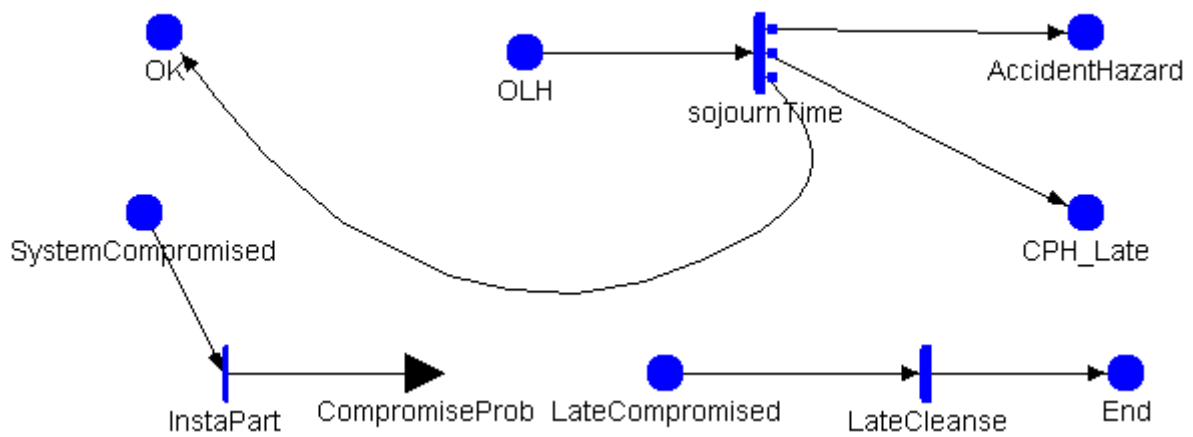


Figure 21: Overlooked Hazard - Defence-in-Depth

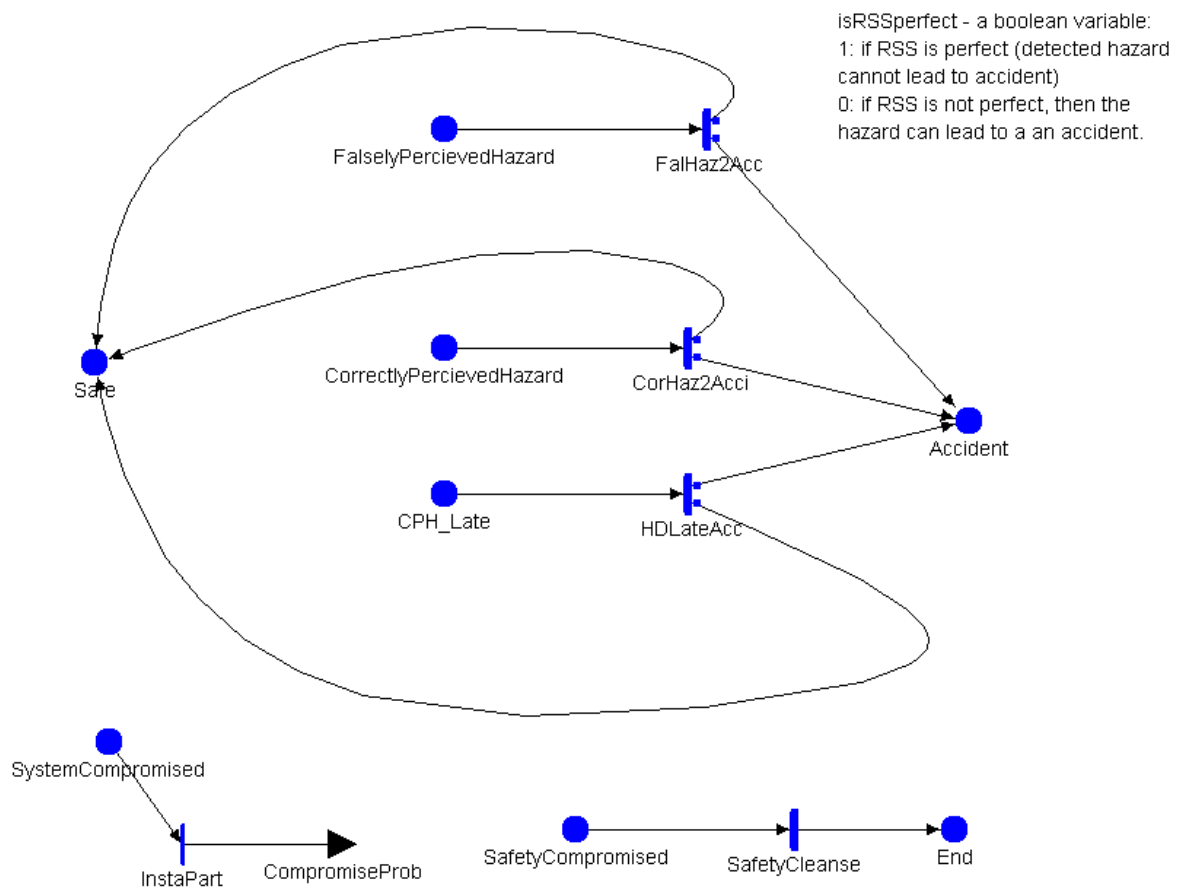


Figure 22: Safety Monitor - Defence-in-Depth

The partitions have now been modified so that the previous ‘Compromised’ state will then lead to an output gate that uses a uniform distribution to generate a random number above or below a chosen probability of successfully compromising a partition.

Each component will also now have its own cleansing rate depending on how much effect there is on safety for the system, to make sure that the attack does not last forever, limiting the duration of the effect.

The layered approach here better represents the vehicle architecture by implementing a layered approach which means the CAN bus is now better taken into consideration compared to before.

Values are similar to before, with the second layer parameters chosen by sensitivity, but another set of parameters is once again from (Ramsbrock, Berthier and Cukier, 2007), which states that when the adversary gets into the system, they were only able to execute commands 22.09% of the time.

6.1.1 Main Model

In the table below, there are now new cleansing and success probability parameters determined through sensitivity analysis. These behave as though the subsystems must now be attacked individually and will cleanse an adversary, suggesting that getting into the system is only the beginning, such as if a CAN bus is successfully accessed.

Parameter	Value(s)
AccidentScale	10.0
CH2AccRate	1.0E-4
CH2OKRate	856.3
DelayScale	1000.0
FH2AccRate	1.0E-5
FH2OKRate	856.3
HazardRate	197.4
HazardScale	10.0
LateCH2AccScale	10.0
MH2AccRate	0.001
MH2CHRate	2250.0
OH2OKRate	1000.0
OH2CH_scale	10.0
attackRate	0.1
attackReset	2.0
attackSuccess	0.2
hazCleanseRate	0.001
hazSuccessProb	0.3
olhCleanseRate	0.001
olhSuccessProb	0.1
safeCleanseRate	0.001
safeSuccessProb	0.5
enableHazard	[0, 1]
enableOLH	[0, 1]
enableSM	[0, 1]
falseHazardProb	1.0E-4
isRSSperfect	0
missHazardProb	0.05

Table 52: Defence-in-Depth Parameters

The values in Table 52 are used as the rates of events and probabilities in the timed activities that make up the SAN model. Apart from the attack and second-layer (i.e. the individual ECU) parameters the rest were obtained from (Popov et al., 2022). The vulnerability modes were assigned in a way to increase the probability of a safety failure by at least one order of magnitude, while the attacks were determined through sensitivity analysis.

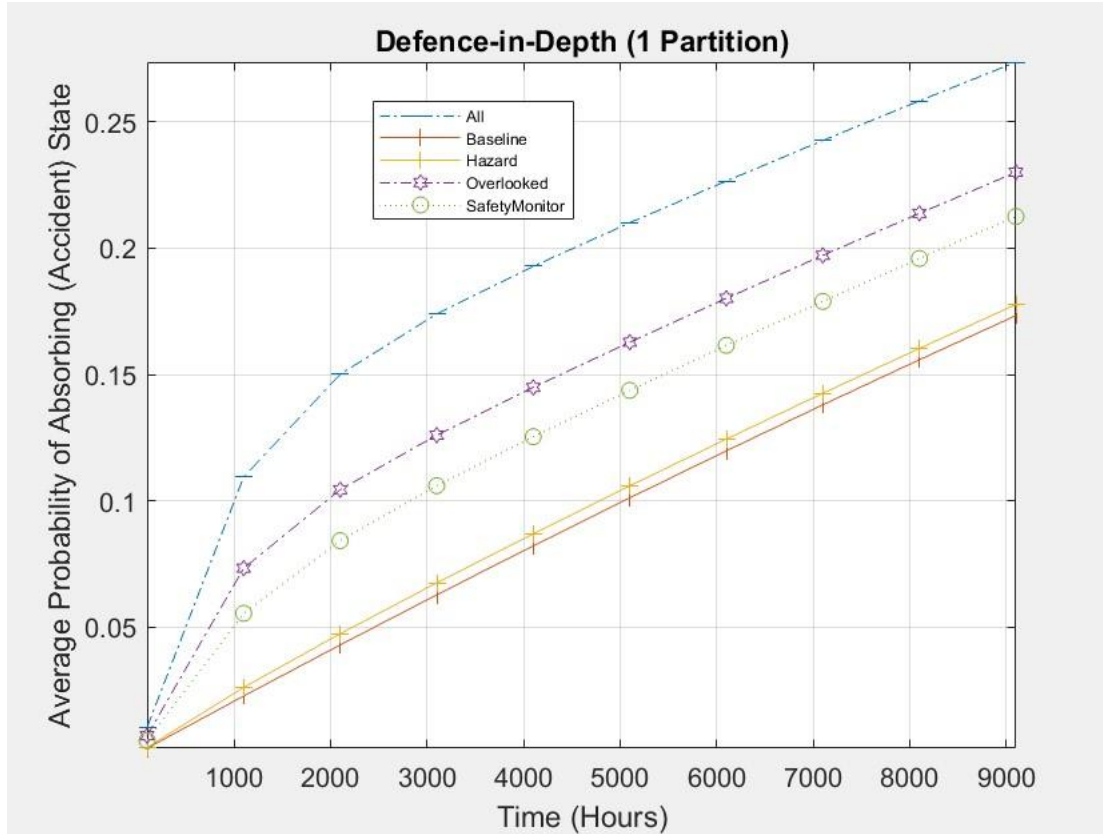


Figure 23: Defence-in-Depth (1 Partition)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
Hazard	0.0024	0.0262	0.0474	0.0675	0.0870	0.1059	0.1245	0.1426	0.1603	0.1777
Overlooked	0.0070	0.0733	0.1044	0.1260	0.1449	0.1628	0.1802	0.1972	0.2138	0.2301
SafetyMonitor	0.0051	0.0555	0.0842	0.1060	0.1254	0.1437	0.1616	0.1789	0.1959	0.2126
All	0.0103	0.1095	0.1501	0.1740	0.1929	0.2101	0.2266	0.2426	0.2583	0.2737

Table 53: Defence-in-Depth - Raw Data (1 Partition)

When defence-in-depth is applied, compromising a single partition as shown in Figure 23 and Table 53 has the following risk of an accident in descending order:

1. Overlooked
2. Safety Monitor
3. Hazard Perception

Therefore, the highest risk in this case is where the component that deals with overlooked hazards at a later point. After this, it is the safety monitor and hazard perception, but the combinations will have more effects, as the compromising of all partitions at once has a greater risk to safety.

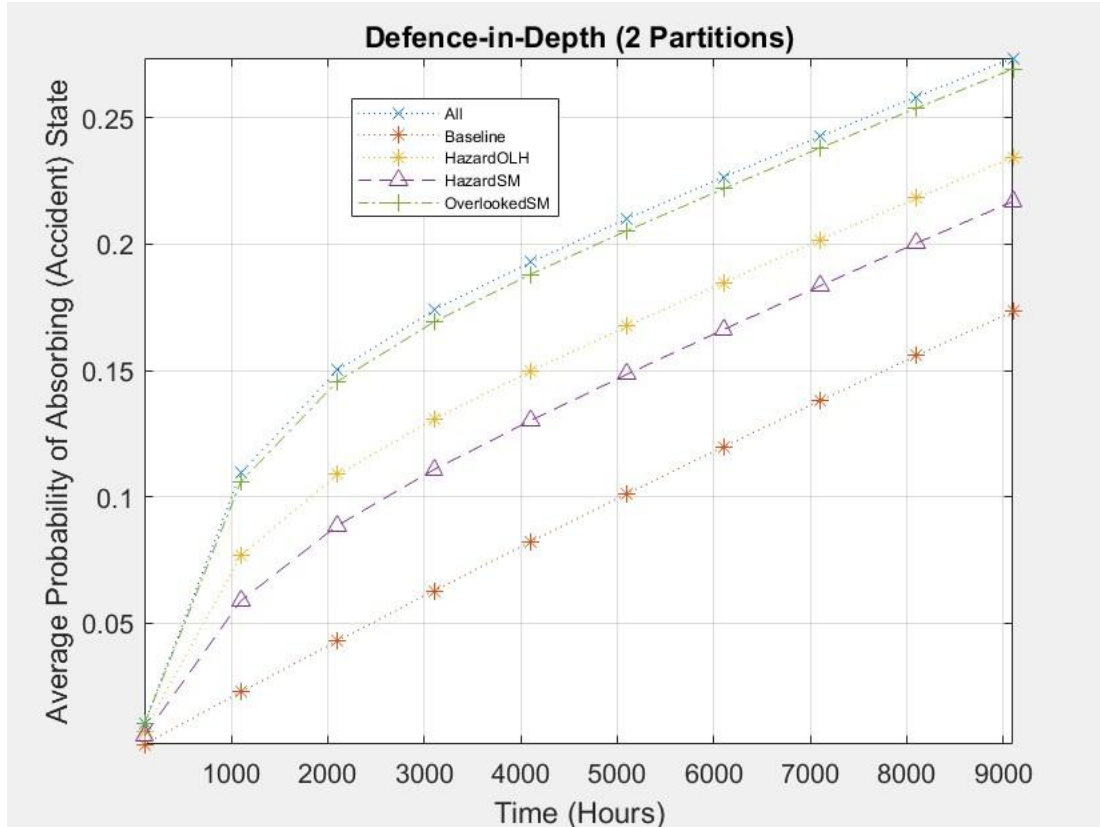


Figure 24: Defence-in-Depth (2 Partitions)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
HazardOLH	0.0073	0.0767	0.1089	0.1308	0.1497	0.1675	0.1848	0.2017	0.2183	0.2345
HazardSM	0.0054	0.0589	0.0887	0.1108	0.1302	0.1485	0.1662	0.1835	0.2004	0.2170
OverlookedSM	0.0100	0.1061	0.1457	0.1692	0.1881	0.2054	0.2219	0.2381	0.2539	0.2693
All	0.0103	0.1095	0.1501	0.1740	0.1929	0.2101	0.2266	0.2426	0.2583	0.2737

Table 54: Defence-in-Depth - Raw Data (2 Partitions)

Looking at the combinations in Figure 24 and Table 54 for combined partitions, the highest to lowest risk to safety is caused by the following:

1. All Partitions
2. Overlooked + Safety Monitor
3. Hazard + Overlooked
4. Hazard + Safety Monitor

Here, unlike in the other models, defence in depth has a difference where instead of the highest risk from compromising two partitions comes from compromising the overlooked hazard and safety monitor. This was in fact the hazard perception and overlooked hazards in the prior models, so the addition of another layer to overcome has changed this, as well as how the safety monitor has greater probability of being successfully compromised, so this is now more likely to be successfully attacked.

6.1.1.1 Discussion

The results from this section indicate the following:

- The most significant individual component in the presence of defence-in-depth is still the late/overlooked hazard partition, suggesting that this is a notable area of concern in vehicles and defences should be applied to this region more than others.
- For two compromised partitions, the late/overlooked hazard perception in combination with the safety monitor is the most significant in this case, as the hazards that are perceived late could lead to accidents which are usually caught by the safety monitor, which is now also not working properly. This change compared to the previous partitioned models without defence-in-depth can be attributed to the higher success probability of attacking the safety monitor which means it may require a stronger defence here.
- The compromising of all partitions is still the worst case here, which makes sense as the whole system would once again not be working properly if this were the case. It could be possible to attempt system-wide defence-in-depth, but this would mean that it would be harder to individually strengthen more vulnerable components.
- There is only a single absorbing state, so the results are limited to one type of accident, when there could be many different scenarios where a safety failure is not necessarily catastrophic, limiting the extent to which the model can assess safety.
- A limitation of the model is that it assumes adversaries that occur at a fixed rate with some success probability, which is typically not 100% meaning that the model does not account for an adversary that can perfectly execute an attack every time without issue (i.e. I assume a limited level of expertise from the adversary).
- It must also be noted that while the model is limited to Exponentially distributed timed activities only, this is not a major issue as according to the findings of (Littlewood, 1979), all Markov models of this type are asymptotically Exponential no matter what distributions I use meaning that my results for the risk would give an Exponential result over time anyway.

6.1.2 Realistic Attack

For this section, the probabilities are all set to 0.2209 based off the statistics in (Ramsbrock, Berthier and Cukier, 2007) where the adversary actually doing something had a probability of 22.09%, so all subsystems will have this as their probability of successful attack, with parameters in the table below. The additional graphs can be found in Appendix 14.3.

Parameter	Value(s)
AccidentScale	10.0
CH2AccRate	1.0E-4
CH2OKRate	856.3
DelayScale	1000.0
FH2AccRate	1.0E-5
FH2OKRate	856.3
HazardRate	197.4
HazardScale	10.0
LateCH2AccScale	10.0
MH2AccRate	0.001
MH2CHRate	2250.0
OH2OKRate	1000.0
OH2CH scale	10.0
attackRate	0.51
attackReset	2.0
attackSuccess	0.0031
hazCleanseRate	0.001
hazSuccessProb	0.2209
olhCleanseRate	0.001
olhSuccessProb	0.2209
safeCleanseRate	0.001
safeSuccessProb	0.2209
enableHazard	[0, 1]
enableOLH	[0, 1]
enableSM	[0, 1]
falseHazardProb	1.0E-4
isRSSperfect	0
missHazardProb	0.05

Table 55: Defence-in-Depth 'Realistic' Parameters

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
Hazard	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
Overlooked	0.0027	0.0546	0.0960	0.1231	0.1440	0.1625	0.1801	0.1972	0.2138	0.2301
SafetyMonitor	0.0024	0.0394	0.0717	0.0963	0.1169	0.1358	0.1539	0.1715	0.1887	0.2055
All	0.0031	0.0712	0.1248	0.1566	0.1787	0.1972	0.2143	0.2307	0.2466	0.2622

Table 56: Defence-in-Depth - 'Realistic' - Raw Data (1 Partition)

For the realistic attack model (Table 56), the risks for compromising a single partition in descending order are as follows:

1. Overlooked Hazard
2. Safety Monitor
3. Hazard Perception

Once again, compromising the component that deals with overlooked hazards (i.e. to correct them later to avoid accidents) is the most serious, though clearly not as serious as a combination attack on the components, as it is still not as severe as compromising all the partitions together.

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
HazardOLH	0.0027	0.0546	0.0960	0.1231	0.1440	0.1625	0.1801	0.1972	0.2138	0.2301
HazardSM	0.0024	0.0394	0.0717	0.0963	0.1169	0.1358	0.1539	0.1715	0.1887	0.2055
OverlookedSM	0.0031	0.0712	0.1248	0.1566	0.1787	0.1972	0.2143	0.2307	0.2466	0.2622
All	0.0031	0.0712	0.1248	0.1566	0.1787	0.1972	0.2143	0.2307	0.2466	0.2622

Table 57: Defence-in-Depth - 'Realistic' - Raw Data (2 Partitions)

For compromising partitions in combination, the risks shown in Table 57 are the following in descending order:

1. All Partitions and OverlookedSM
2. Hazard + Overlooked

In this case, a 'realistic' attack shows that compromising the overlooked hazards and safety monitor are equal in terms of risk, meaning that compromising the hazard perception is not even necessary in this case due to its limited effect on safety. Therefore, these are the two most important components in the CAV to defend for this architecture.

6.1.2.1 Discussion

The results from this section indicate the following:

- The most significant individual component in the presence of defence-in-depth is still the late/overlooked hazard partition, suggesting that this is a notable area of concern in vehicles and defences should be applied to this region more than others.
- For two compromised partitions, the late/overlooked hazard perception in combination with the safety monitor is the most significant in this case, as the hazards that are perceived late could lead to accidents which are usually caught by the safety monitor, which is now also not working properly. This change compared to the previous partitioned models without defence-in-depth can be attributed to the higher success probability of attacking the safety monitor which means it may require a stronger defence here.
- The compromising of all partitions is still the worst case here, which makes sense as the whole system would once again not be working properly if this were the case. While a system-wide approach could mitigate this, it would not allow individual securing of more vulnerable components.
- There is only a single absorbing state, so the results are limited to one type of accident, when there could be many different scenarios where a safety failure is not necessarily catastrophic, limiting the extent to which the model can assess safety.
- A limitation of the model is that it assumes adversaries that occur at a fixed rate with some success probability, which is typically not 100% meaning that the model does not account for an adversary that can perfectly execute an attack every time without issue (i.e. I assume a limited level of expertise from the adversary).
- It must also be noted that while the model is limited to Exponentially distributed timed activities only, this is not a major issue as according to the findings of (Littlewood, 1979), all Markov models of this type are asymptotically Exponential no matter what distribution I use meaning that my results for the risk would give an Exponential result over time anyway.

6.1.3 Weak vs. Strong Adversary

To see the effective range of defence-in-depth, it is important to have a comparison of strong and weak adversaries. This is done by having an adversary with low rate of attack and low attack success probability, as well as one with high attack success probability and high rate of attack. The parameterisation for this comparison is shown in the table below.

Parameter	Weak	Strong
AccidentScale	10.0	10.0
CH2AccRate	1.0E-4	1.0E-4
CH2OKRate	856.3	856.3
DelayScale	1000.0	1000.0
FH2AccRate	1.0E-5	1.0E-5
FH2OKRate	856.3	856.3
HazardRate	197.4	197.4
HazardScale	10.0	10.0
LateCH2AccScale	10.0	10.0
MH2AccRate	0.001	0.001
MH2CHRate	2250.0	2250.0
OH2OKRate	1000.0	1000.0
OH2CH scale	10.0	10.0
attackRate	0.1	0.3
attackReset	2.0	2.0
attackSuccess	0.25	0.75
hazCleanseRate	0.001	0.001
hazSuccessProb	0.3	0.3
olhCleanseRate	0.001	0.001
olhSuccessProb	0.1	0.1
safeCleanseRate	0.001	0.001
safeSuccessProb	0.5	0.5
enableHazard	[0, 1]	[0, 1]
enableOLH	[0, 1]	[0, 1]
enableSM	[0, 1]	[0, 1]
falseHazardProb	1.0E-4	1.0E-4
isRSSperfect	0	0
missHazardProb	0.05	0.05

Table 58: Defence-in-Depth - Weak & Strong Attack Parameters

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
StrongAll	0.0158	0.1113	0.1506	0.1741	0.1929	0.2101	0.2266	0.2426	0.2583	0.2737
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
WeakAll	0.0113	0.1099	0.1502	0.1740	0.1929	0.2101	0.2266	0.2426	0.2583	0.2737

Table 59: Weak vs. Strong - All Partitions

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
StrongHazard	0.0026	0.0262	0.0475	0.0675	0.0870	0.1059	0.1245	0.1426	0.1603	0.1777
WeakHazard	0.0024	0.0262	0.0474	0.0675	0.0870	0.1059	0.1245	0.1426	0.1603	0.1777

Table 60: Weak vs. Strong - Hazard Perception

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
StrongHazOLH	0.0108	0.0778	0.1092	0.1308	0.1497	0.1675	0.1848	0.2017	0.2183	0.2345
WeakHazOLH	0.0075	0.0770	0.1089	0.1308	0.1497	0.1675	0.1848	0.2017	0.2183	0.2345

Table 61: Weak vs. Strong - Hazard + Overlooked

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
StrongHazSM	0.0076	0.0597	0.0889	0.1108	0.1302	0.1485	0.1662	0.1835	0.2004	0.2170
WeakHazSM	0.0058	0.0591	0.0887	0.1108	0.1302	0.1485	0.1662	0.1835	0.2004	0.2170

Table 62: Weak vs. Strong - Hazard + Safety Monitor

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
StrongOLH	0.0103	0.0743	0.1047	0.1261	0.1449	0.1628	0.1802	0.1972	0.2138	0.2301
WeakOLH	0.0076	0.0736	0.1045	0.1260	0.1449	0.1628	0.1802	0.1972	0.2138	0.2301

Table 63: Weak vs. Strong – Overlooked Hazard

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
StrongSafeOLH	0.0153	0.1078	0.1462	0.1694	0.1881	0.2054	0.2219	0.2381	0.2539	0.2693
WeakSafeOLH	0.0109	0.1065	0.1458	0.1693	0.1881	0.2054	0.2219	0.2381	0.2539	0.2693

Table 64: Weak vs. Strong - Overlooked + Safety Monitor

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
StrongSafety	0.0071	0.0562	0.0845	0.1061	0.1254	0.1438	0.1616	0.1789	0.1959	0.2126
WeakSafety	0.0054	0.0556	0.0843	0.1060	0.1254	0.1438	0.1616	0.1789	0.1959	0.2126

Table 65: Weak vs. Strong - Safety Monitor

For this section, looking at the results from Table 59-Table 65, it can be seen that the effects of the weak and strong adversary on a system using defence-in-depth are quite close to one another, meaning defence in depth works well against both.

6.1.3.1 Discussion

The results of comparing adversaries for defence-in-depth have given the following conclusions:

- Attacks on different partitions for both weak and strong adversaries give the same risk to safety in both cases.
- The risks of different attacks relative to one another has remained unchanged with the changing of the adversarial strength.

6.1.4 Sensitivity Analysis

To choose appropriate parameters, sensitivity analysis was used where empirical datasets or existing papers did not offer it, which meant using ranges of acceptable or plausible values to determine the closest thing to a suitable value in the SAN models, as shown in the table below. When varying the cleansing, success probabilities were set to 1.0 to assume definite success so that the values could be chosen according to the worst-case scenario of an adversary attacking the partitions, with the cleansing parameters of other components being kept at 0.001 to limit the effect of the cleansing in another part of the system.

Parameter	Value(s)
AccidentScale	10.0
CH2AccRate	1.0E-4
CH2OKRate	856.3
DelayScale	1000.0
FH2AccRate	1.0E-5
FH2OKRate	856.3
HazardRate	197.4
HazardScale	10.0
LateCH2AccScale	10.0
MH2AccRate	0.001
MH2CHRate	2250.0
OH2OKRate	1000.0
OH2CH scale	10.0
attackRate	0.1
attackReset	[0.001,0.01,0.1,1.0,2.0,4.0]
attackSuccess	0.2
hazCleanseRate	[0.001,0.01,0.1,1.0,2.0,4.0]
hazSuccessProb	[0.1,0.3,0.5]
olhCleanseRate	[0.001,0.01,0.1,1.0,2.0,4.0]
olhSuccessProb	[0.1,0.3,0.5]
safeCleanseRate	[0.001,0.01,0.1,1.0,2.0,4.0]
safeSuccessProb	[0.1,0.3,0.5]
enableHazard	[0, 1]
enableOLH	[0, 1]
enableSM	[0, 1]
falseHazardProb	1.0E-4
isRSSperfect	0
missHazardProb	0.05

Table 66: Sensitivity Analysis Parameters - Defence-in-Depth

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
Reset=0.001	0.0115	0.1218	0.1652	0.1896	0.2083	0.2253	0.2415	0.2573	0.2726	0.2877
Reset=0.01	0.0115	0.1218	0.1652	0.1896	0.2083	0.2253	0.2415	0.2573	0.2726	0.2877
Reset=0.1	0.0115	0.1218	0.1652	0.1896	0.2083	0.2253	0.2415	0.2573	0.2726	0.2877
Reset=1.0	0.0115	0.1218	0.1652	0.1896	0.2083	0.2253	0.2415	0.2573	0.2726	0.2877
Reset=2.0	0.0115	0.1218	0.1652	0.1896	0.2083	0.2253	0.2415	0.2573	0.2726	0.2877
Reset=4.0	0.0115	0.1218	0.1652	0.1896	0.2083	0.2253	0.2415	0.2573	0.2726	0.2877

Table 67: Defence-in-Depth - Attack Reset Sensitivity (Raw Data)

For the attack reset rates shown in Table 67, there is very little difference for the attack reset on the system-wide level, so any of these could have been used. However, to remain consistent on this level, the rate of 2.0 was maintained as it was in line with this and the previous models.

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
Cleansing=0.001	0.0115	0.1218	0.1652	0.1896	0.2083	0.2253	0.2415	0.2573	0.2726	0.2877
Cleansing=0.01	0.0114	0.1183	0.1604	0.1844	0.2031	0.2201	0.2364	0.2523	0.2677	0.2829
Cleansing=0.1	0.0112	0.1178	0.1599	0.1838	0.2026	0.2196	0.2359	0.2517	0.2672	0.2824
Cleansing=1.0	0.0112	0.1177	0.1598	0.1838	0.2025	0.2196	0.2359	0.2517	0.2672	0.2824
Cleansing=2.0	0.0112	0.1177	0.1598	0.1838	0.2025	0.2196	0.2359	0.2517	0.2672	0.2824
Cleansing=4.0	0.0112	0.1177	0.1598	0.1838	0.2025	0.2196	0.2359	0.2517	0.2672	0.2824

Table 68: Defence-in-Depth - Hazard Cleansing Sensitivity (Raw Data)

In this part (Table 68), the hazard perception subsystem cleansing works as expected, with higher rates lowering the risk of accident as the window of time for which the attack has an effect is reduced significantly. Therefore, choosing the highest would have the best effect. However, this was found to conceal the actual effects of attacks on a system using defence-in-depth so a cleansing rate of 0.001 was used so that an attack would last at least one thousand hours if successful.

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
Cleansing=0.001	0.0115	0.1218	0.1652	0.1896	0.2083	0.2253	0.2415	0.2573	0.2726	0.2877
Cleansing=0.01	0.0099	0.0703	0.1001	0.1220	0.1413	0.1594	0.1769	0.1939	0.2106	0.2270
Cleansing=0.1	0.0064	0.0607	0.0907	0.1129	0.1323	0.1505	0.1682	0.1855	0.2024	0.2189
Cleansing=1.0	0.0055	0.0597	0.0898	0.1119	0.1313	0.1496	0.1673	0.1846	0.2015	0.2180
Cleansing=2.0	0.0055	0.0597	0.0897	0.1119	0.1313	0.1496	0.1673	0.1846	0.2014	0.2180
Cleansing=4.0	0.0055	0.0597	0.0897	0.1119	0.1313	0.1496	0.1673	0.1846	0.2014	0.2180

Table 69: Defence-in-Depth - Overlooked Cleansing Sensitivity (Raw Data)

In this part (Table 69), the overlooked hazard subsystem cleansing works as expected, with higher rates lowering the risk of accident as the window of time for which the attack has an effect is reduced significantly. Therefore, choosing the highest would have the best effect. However, this was found to conceal the actual effects of attacks on a system using defence-in-depth so a cleansing rate of 0.001 was used so that an attack would last at least one thousand hours if successful.

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
Cleansing=0.001	0.0115	0.1218	0.1652	0.1896	0.2083	0.2253	0.2415	0.2573	0.2726	0.2877
Cleansing=0.01	0.0107	0.0944	0.1292	0.1515	0.1702	0.1876	0.2045	0.2210	0.2372	0.2530
Cleansing=0.1	0.0090	0.0897	0.1245	0.1469	0.1657	0.1832	0.2002	0.2168	0.2330	0.2489
Cleansing=1.0	0.0086	0.0892	0.1240	0.1464	0.1652	0.1828	0.1998	0.2164	0.2326	0.2485
Cleansing=2.0	0.0086	0.0891	0.1240	0.1464	0.1652	0.1828	0.1998	0.2164	0.2326	0.2485
Cleansing=4.0	0.0085	0.0891	0.1240	0.1464	0.1652	0.1828	0.1998	0.2164	0.2326	0.2485

Table 70: Defence-in-Depth - Safety Cleansing Sensitivity (Raw Data)

In this part (Table 70), the safety monitor cleansing works as expected, with higher rates lowering the risk of accident as the window of time for which the attack has an effect is reduced significantly. Therefore, choosing the highest would have the best effect. However, this was found to conceal the actual effects of attacks on a system using defence-in-depth so a cleansing rate of 0.001 was used so that an attack would last at least one thousand hours if successful.

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
Success=0.1	0.0112	0.1177	0.1598	0.1838	0.2025	0.2195	0.2358	0.2517	0.2672	0.2824
Success=0.3	0.0115	0.1211	0.1643	0.1885	0.2073	0.2243	0.2405	0.2562	0.2717	0.2867
Success=0.5	0.0115	0.1211	0.1643	0.1885	0.2073	0.2243	0.2405	0.2562	0.2717	0.2867

Table 71: Defence-in-Depth - Hazard Success Sensitivity (Raw Data)

For the probability of successfully compromising the hazard perception subsystem (Table 71), the sensitivity analysis indicated the obvious, that higher success causes a greater risk of accident. The success probability risks are as follows in descending order:

1. Success=0.3 and 0.5
2. Success=0.1

Therefore, to be safe, values of 0.3 and 0.5 are suitable, so to use the lowest success rate with good safety, the value of 0.3 as a success rate for compromising the hazard perception in the model.

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
Success=0.1	0.0104	0.1102	0.1511	0.1750	0.1939	0.2111	0.2276	0.2437	0.2593	0.2747
Success=0.3	0.0104	0.1102	0.1511	0.1750	0.1939	0.2111	0.2276	0.2437	0.2593	0.2747
Success=0.5	0.0105	0.1108	0.1518	0.1757	0.1946	0.2118	0.2283	0.2443	0.2600	0.2753

Table 72: Defence-in-Depth - Overlooked Success Sensitivity (Raw Data)

For the probability of successfully compromising the overlooked hazard subsystem (Table 72) the sensitivity analysis indicated the obvious, that higher success causes a greater risk of accident. The success probability risks are as follows in descending order:

1. Success=0.5
2. Success=0.3
3. Success=0.1

Therefore, while aiming not to have an unusually high success probability, the other success probabilities had similar risks of accident, so a success probability of 0.1 was chosen for the attack on the overlooked partition.

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
Success=0.1	0.0110	0.1159	0.1578	0.1818	0.2005	0.2176	0.2340	0.2499	0.2654	0.2806
Success=0.3	0.0110	0.1159	0.1578	0.1818	0.2005	0.2176	0.2340	0.2499	0.2654	0.2806
Success=0.5	0.0115	0.1218	0.1652	0.1896	0.2083	0.2253	0.2415	0.2573	0.2726	0.2877

Table 73: Defence-in-Depth - Safety Success Sensitivity (Raw Data)

For the probability of successfully compromising the safety monitor (Table 73) the sensitivity analysis indicated the obvious, that higher success causes a greater risk of accident. The success probability risks are as follows in descending order:

1. Success=0.5
2. Success=0.3 and 0.1

Any value of 0.1 or 0.3 would be suitable, but the differences were negligible, and a worst-case scenario seemed suitable due to the lack of actual effect of compromising the safety monitor in prior models meant that 0.5 was chosen as the success probability.

6.2 Intrusion Tolerance

This section looks at a different form of defence going beyond intrusion detection and tolerance, instead assuming the eventual and inevitable success of an adversary. Intrusion tolerance focuses on the preservation of the functional properties of the system behaviour and the response of the system to such a compromise (Section 2.7.3) when the adversary does eventually get in, by having replicas of each of the system components of which a certain amount must be overcome to cause the system to be truly compromised or failed.

For a value m ($m > 0$), the number of replicas must be set at $3m+1$ in accordance with the Byzantine agreement which states this to be the number of participants for which $m+1$ replicas must be overcome to compromise the component in question (e.g. for $m=1$, there will be 4 replicas of which at least 2 must be compromised/crashed to cause the component to fail). The second part of intrusion tolerance aside from the replicas, is that these replicas are periodically cleansed, so an adversary must overcome $m+1$ replicas faster than the system is able to cleanse them to compromise security.

In this type of model, the intrusion tolerance is where the attacks are directed and only once the failed state (i.e. $m+1$ participants have been compromised) is reached will the ‘Compromised’ state in the main component (i.e. the main partition such as perception, which is the same as in the original partitioned models) become active and have an actual effect on safety. The attack model is also different to all previous model, attacking each partition at an individual rate and success probability. By having this layered approach as well as parallel attacks on the components and their replicas, my modelling approach assumes diversification in the replicas.

The later models also become dynamic, adaptively changing the number of replicas, as well as introducing varying cleansing rates and attack rates to model the possibility of adversaries with different capabilities.

Experiment	percEnable	sojEnable	smEnable
Baseline	0	0	0
Hazard	1	0	0
Overlooked	0	1	0
SafetyMonitor	0	0	1
HazardOLH	1	1	0
HazardSM	1	0	1
OverlookedSM	0	1	1
All	1	1	1

Table 74: Intrusion Tolerance - Attack Combination Table

It can be seen in Table 74 that for the same experiment in the previous partitioned models, which should be interpreted as the different possible combinations of partitions which can be attacked in the vehicle, with 1 representing an enabled attack where 0 represents the opposite. The names of the flags used to enable or disable the effect of an adversary were renamed in the process to the following equivalents:

1. Hazard = percEnable
2. Overlooked = sojEnable
3. SafetyMonitor = smEnable

In addition to this, the relevant cleansing and replica parameters that are added to reflect the intrusion tolerance have been added and will be reflected in the tables for each probabilistic model. It must be noted that the attacks are against partitions separately and in combination

as opposed to a system-wide attack for this model, so compromising a partition does not guarantee compromising all others. The full model documentation can be found in Appendix 13.9. It should be noted that due to the use of Monte Carlo, the information is presented as a figure followed by the raw data for the risk of accident immediately followed by the confidence levels for the results in question.

6.2.1 Composed Models

The below composed models are the primary structures for the models, with changes to the attack model and intrusion tolerance where needed. The models have fixed attacks and fixed replicas, followed by dynamic replication to which dynamic adversaries and cleansing rates are added in the form of a cycle to see how the system is affected. As intrusion tolerance has a number of replicas that can be varied, as well as different levels in order to dynamically change replicas or other values, extended places (represented by an orange circle) are used to achieve this, as they can act as arrays or structures, which is preferable to creating many regular places (represented by a blue circle) that would overcomplicate the system by creating too many states.

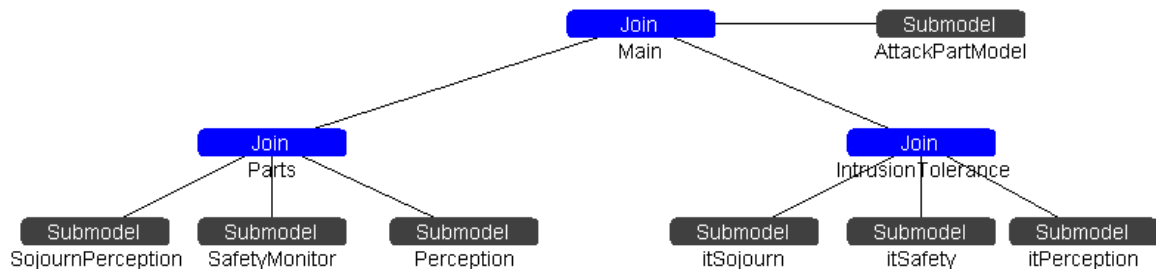


Figure 25: Fixed Intrusion Tolerance

In Figure 25, the sections can be described as follows:

1. **AttackPartModel:** An attack with separate branches for attacking each partition, which will only have effect if the attack is successful, and the partition is ‘enabled’ to allow attacks to take effect.
2. **itSojourn:** Intrusion tolerance module applied to the Sojourn/Overlooked Hazards.
3. **itSafety:** Intrusion tolerance module applied to the Safety Monitor.
4. **itPerception:** Intrusion tolerance module applied to the main Hazard Perception.
5. **SojournPerception:** Module for perception of hazards that may have been initially overlooked.
6. **SafetyMonitor:** Vehicle safety monitor designed to make sure the vehicle behaves safely to limit the probability of accidents by preventing dangerous driving.
7. **Perception:** Main module for the perception of hazards.

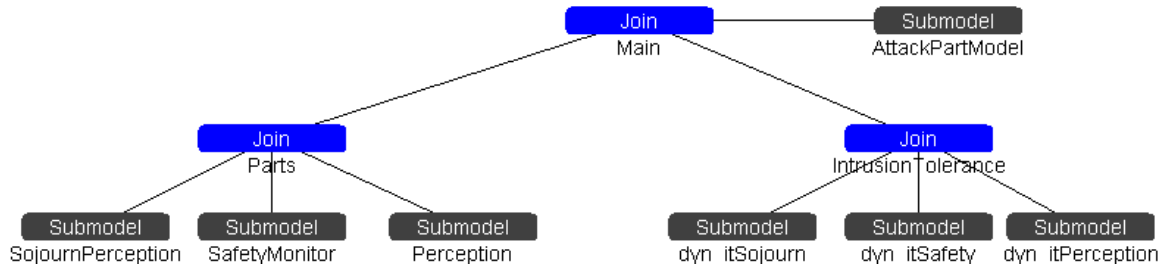


Figure 26: Dynamic Intrusion Tolerance

In Figure 26, the sections can be described as follows:

1. **AttackPartModel:** An attack with separate branches for attacking each partition, which will only have effect if the attack is successful, and the partition is ‘enabled’ to allow attacks to take effect.
2. **dyn_itSojourn:** Adaptive intrusion tolerance module applied to the Sojourn/Overlooked Hazards.
3. **dyn_itSafety:** Adaptive intrusion tolerance module applied to the Safety Monitor.
4. **dyn_itPerception:** Adaptive intrusion tolerance module applied to the main Hazard Perception.
5. **SojournPerception:** Module for perception of hazards that may have been initially overlooked.
6. **SafetyMonitor:** Vehicle safety monitor designed to make sure the vehicle behaves safely to limit the probability of accidents by preventing dangerous driving.
7. **Perception:** Main module for the perception of hazards.

By having the attacks followed by the replicas which need to be compromised for $m+1$ amount, this introduces a layered approach which means each attack now passes through the CAN bus and must then overcome the intrusion tolerant property of each component.

The attack model can also become dynamic to make it more difficult to deal with and observe a range of attacks rather than one fixed adversary, as well as the possibility of dynamic cleansing rates as opposed to just replication. The purpose of the fixed model is to look at intrusion tolerance in its standard form and view the effectiveness while dynamic models look at how an adaptive approach would work as well as creating the more unpredictable environment that would be expected for a vehicle on the road.

6.2.2 Attacks

There are two attack models in use for the intrusion tolerance in these cases. There is a fixed attack model where all rates are fixed, which is used for the fixed model and the dynamic replica model, as well as a dynamic attack model (i.e. attack rates changing in intensity from lowest to highest, with a rate of 1 in 1000 hours). The purpose of these models is as follows:

1. **Fixed:** A fixed number of replicas and cleansing with a fixed adversary allow the effectiveness of intrusion tolerance to be observed against a single type of adversary of non-varying ability.
2. **Dynamic:** By introducing different dynamic elements, the environment can be made less predictable. Dynamic replication allows showing how an adaptive model which increases or decreases replicas as a response will affect the safety of the system.

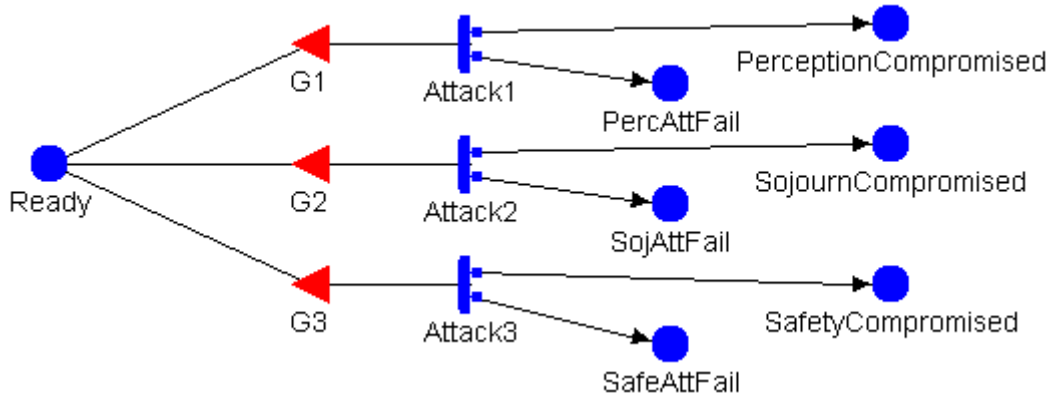


Figure 27: Intrusion Tolerance Attack Model

In Figure 27, a new attack model is introduced where each partition can be attacked individually with its own fixed attack intensity and probability. This is to account for the possibility that while all partitions can be attacked, a combination could succeed partially, so there is no assumption that an attack compromises both successfully, and that their vulnerabilities may depend on one another entirely.

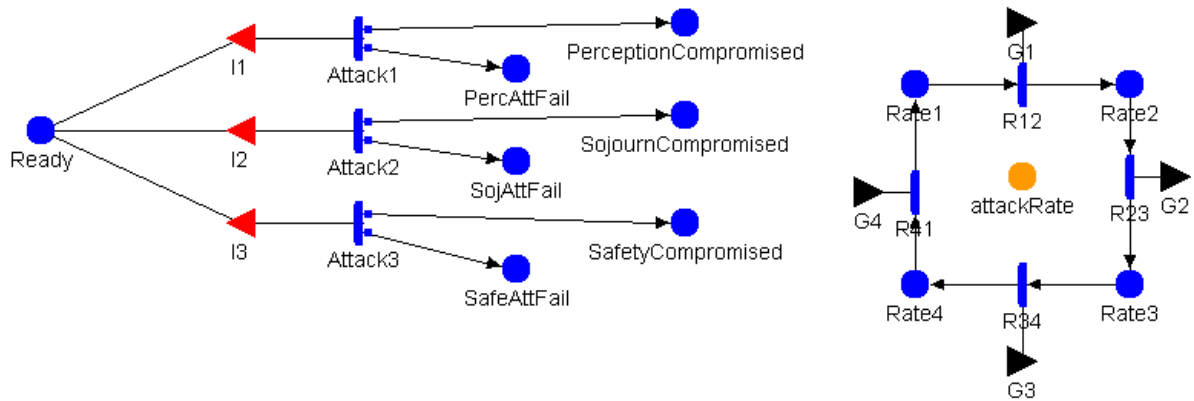


Figure 28: Intrusion Tolerance Attack Model - Dynamic Attack Rates

For Figure 28, the attacks are now not only separated, but their intensities vary at different rates (in this case every 1000 hours), making use of extended places (represented by an orange circle) to have a varying attack rate that is fed to the timed activities. These mean that the adversary is now less predictable and will gradually go to a high intensity before starting over, which means the system is now no longer trying to deal with an adversary of fixed strength.

6.2.3 Intrusion Tolerant Components

Here in the figures and tables below, it is seen how intrusion tolerance is applied one stage before the components, so that an adversary must successfully overcome this mechanism to introduce a failed state before being able to fully compromise a component. These use dynamic and static components depending on the model in use.

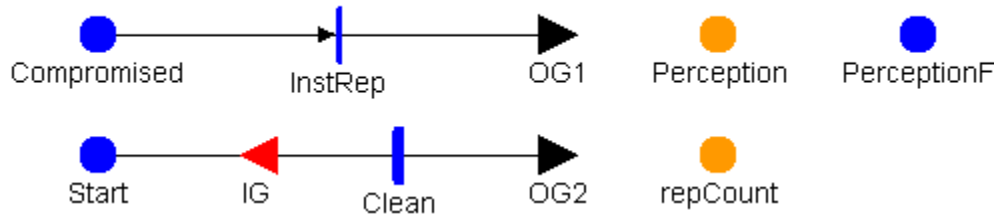


Figure 29: Intrusion Tolerance Mechanism – Perception

This first model (Figure 29) is straightforward, and can be broken down into the following:

1. **Compromised:** The state reached when a replica is compromised, which can bring the system closer to failure.
2. **Start:** The start state of the cleansing, which is always enabled.
3. **Clean:** Timed activity for periodically cleansing the replicas.
4. **OG1:** The output gate designed to immediately compromise a replica once an adversary has succeeded. It will stay compromised until it is cleansed. Also checks to see if enough replicas have been overcome to move into the failed state.
5. **OG2:** The output gate designed to set a replica in the cleansing cycle back to normal by removing the compromised state once it is clean.
6. **IG:** Input gate which makes sure to check that the cleansing cycle is enabled through 'Start' and makes sure the token never sets to 0 on its own so the cycle is not inadvertently turned off.
7. **Perception:** Extended place holding an array the size of the number of replicas, each of which is set to 1 when compromised and will only be set to 0 when the cleansing mechanism is applied.
8. **repCount:** This extended place counts the current replica position in the cleansing cycle (i.e. if you are at the 1st, 2nd, 3rd, or 4th replica etc.).
9. **PerceptionF:** The failed state for the module, where the number of replicas compromised is above the threshold of the Byzantine agreement, so the system can no longer function in a safe or secure manner.

The other component intrusion tolerances for the fixed model exist in a similar manner, with slight name changes to reflect that they are a different component but use the same mechanisms.

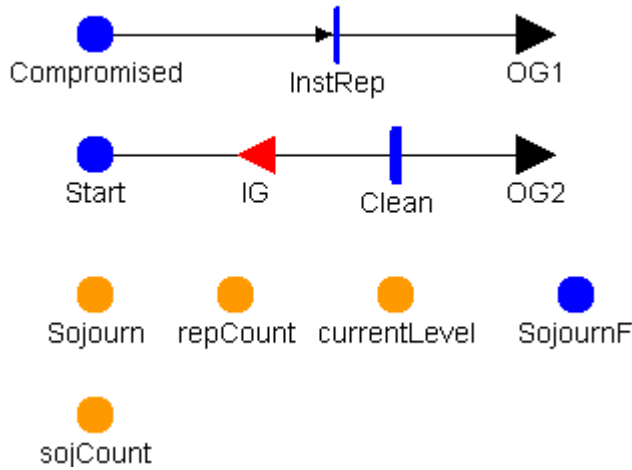


Figure 30: Dynamic Intrusion Tolerance Mechanism – Overlooked Hazard

This model in Figure 30 is more dynamic can be broken down into the following:

1. **Compromised:** The state reached when a replica is compromised, which can bring the system closer to failure.
2. **Start:** The start state of the cleansing, which is always enabled.
3. **Clean:** Timed activity for periodically cleansing the replicas.
4. **OG1:** The output gate designed to immediately compromise a replica once an adversary has succeeded. It will stay compromised until it is cleansed. Also checks to see if enough replicas have been overcome to move into the failed state.
5. **OG2:** The output gate designed to set a replica in the cleansing cycle back to normal by removing the compromised state once it is clean. Checking to see if the replicas are failed, this also checks if the number of replicas cleansed exceed the threshold for failure or not, adaptively increasing or decreasing the number of replicas.
6. **IG1:** Input gate which makes sure to check that the cleansing cycle is enabled through 'Start' and makes sure the token never sets to 0 on its own so the cycle is not inadvertently turned off.
7. **Sojourn:** Extended place holding an array the size of the number of replicas, each of which is set to 1 when compromised and will only be set to 0 when the cleansing mechanism is applied.
8. **repCount:** This extended place counts the current replica position in the cleansing cycle (i.e. if you are at the 1st, 2nd, 3rd, or 4th replica etc.).
9. **sojCount:** Extended place to check the replica being cleansed, but additional due to the dynamic nature of this model, typically used to check for the threshold as well.
10. **currentLevel:** As the model is dynamic, this is the base used to determine the current number of replicas based on the increase or decrease in levels (i.e. if the level is 1 then the replicas will be 4 as per Byzantine agreement on participants).
11. **SojournF:** The failed state for the module, where the number of replicas compromised is above the threshold of the Byzantine agreement, so the system can no longer function in a safe or secure manner.

The dynamic replication here means that intrusion tolerance is more adaptive, so an adversary that tries to overcome the number of replicas needed to induce failure will have a harder time,

with resources being allocated or taken away according to need, so more replicas mean a higher threshold where the attacker is becoming too successful.

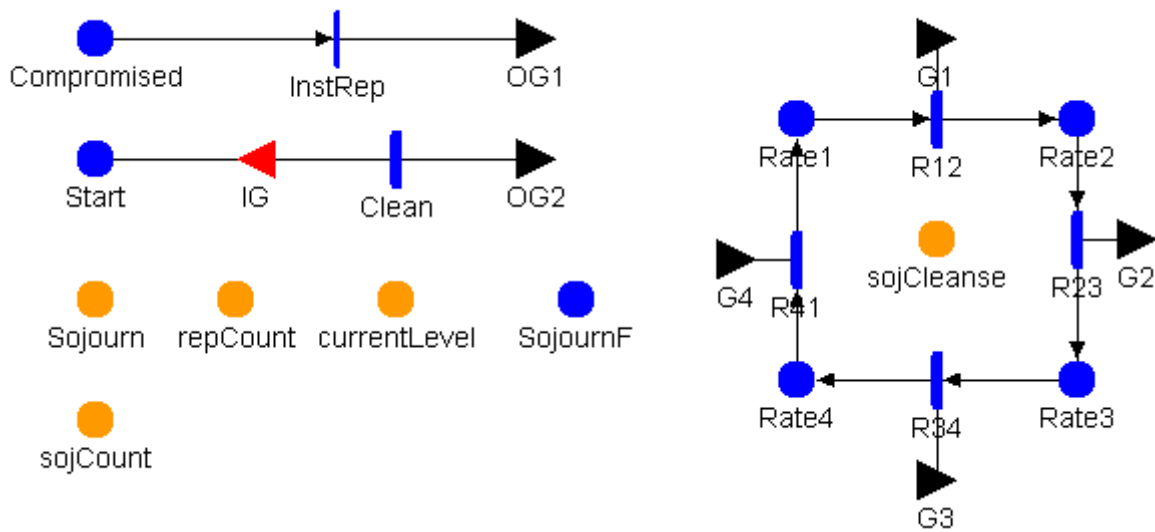


Figure 31: Dynamic Intrusion Tolerance Mechanism – Overlooked Hazard (Dynamic Cleansing)

The model in Figure 31 can be broken down into the following:

1. **Compromised:** The state reached when a replica is compromised, which can bring the system closer to failure.
2. **Start:** The start state of the cleansing, which is always enabled.
3. **Clean:** Timed activity for periodically cleansing the replicas.
4. **OG1:** The output gate designed to immediately compromise a replica once an adversary has succeeded. It will stay compromised until it is cleansed. Also checks to see if enough replicas have been overcome to move into the failed state.
5. **OG2:** The output gate designed to set a replica in the cleansing cycle back to normal by removing the compromised state once it is clean. Checking to see if the replicas are failed, this also checks if the number of replicas cleansed exceed the threshold for failure or not, adaptively increasing or decreasing the number of replicas.
6. **IG1:** Input gate which makes sure to check that the cleansing cycle is enabled through 'Start' and makes sure the token never sets to 0 on its own so the cycle is not inadvertently turned off.
7. **Sojourn:** Extended place holding an array the size of the number of replicas, each of which is set to 1 when compromised and will only be set to 0 when the cleansing mechanism is applied.
8. **repCount:** This extended place counts the current replica position in the cleansing cycle (i.e. if you are at the 1st, 2nd, 3rd, or 4th replica etc.).
9. **sojCount:** Extended place to check the replica being cleansed, but additional due to the dynamic nature of this model, typically used to check for the threshold as well.
10. **sojCleanse:** Extended place to store the most recent cleansing rate to feed to the timed activity for cleansing.
11. **R12, R23, R34, R41:** Timed activities for the increase or decrease in the cleansing rate.

12. **G1,G2,G3,G4:** The output gates which will set the new cleansing rate at the extended place.
13. **currentLevel:** As the model is dynamic, this is the base used to determine the current number of replicas based on the increase or decrease in levels (i.e. if the level is 1 then the replicas will be 4 as per Byzantine agreement on participants).
14. **SojournF:** The failed state for the module, where the number of replicas compromised is above the threshold of the Byzantine agreement, so the system can no longer function in a safe or secure manner.

These dynamic variations introduce less rigid cleansing to counteract the strength of a dynamic adversary, in this case set to the same rates at which the adversary may vary their behaviour, so that a strong adversary could be prevented faster by having equally intense cleansing to give them less time to have an effect.

6.2.4 Fixed Model

This section has the original intrusion tolerance model, which is completely static with a fixed set of attack parameters, using four replicas and the same cleansing rate throughout. The table below shows that this is all applied to each partition individually as opposed to the system, which is more complex than in other models which were all using a single attack which would simply attack all partitions at once.

The attack is fixed, and even when enabled, the probability of success means that each partition has separate success and failure. The models were simulated across multiple points in time from 100-9100 hours in 1000-hour steps, using Monte Carlo simulations for 50000 batches in steps of 1000 batches to observe the change in accident probability over the period for which the CAV is observed.

Parameter	Value(s)
AccidentScale	10.0
CH2AccRate	1.0E-4
CH2OKRate	856.3
DelayScale	1000.0
FH2AccRate	1.0E-5
FH2OKRate	856.3
HazardRate	197.4
HazardScale	10.0
LateCH2AccScale	10.0
MH2AccRate	0.001
MH2CHRate	2250.0
OH2OKRate	1000.0
OH2CH scale	10.0
percCleanse	0.3
percReps	4
percRate	0.1
percSuccess	0.2
sojCleanse	0.3
sojReps	4
sojRate	0.1
sojSuccess	0.2
smCleanse	0.3
smReps	4
smRate	0.1
smSuccess	0.2
percEnable	[0, 1]
sojEnable	[0, 1]
smEnable	[0, 1]
falseHazardProb	1.0E-4
isRSSperfect	0
missHazardProb	0.05

Table 75: Intrusion Tolerance Parameters - Static Model

The purpose of this model is to show intrusion tolerance effectiveness as a form of defence where all partitions have equal replicas and cleansing rates (4 and 0.3 respectively). It must be noted that this is not a system wide attack, as the attacks branch out for each partition, with separate rates and success probabilities. All parameters for this model are shown in Table 75.

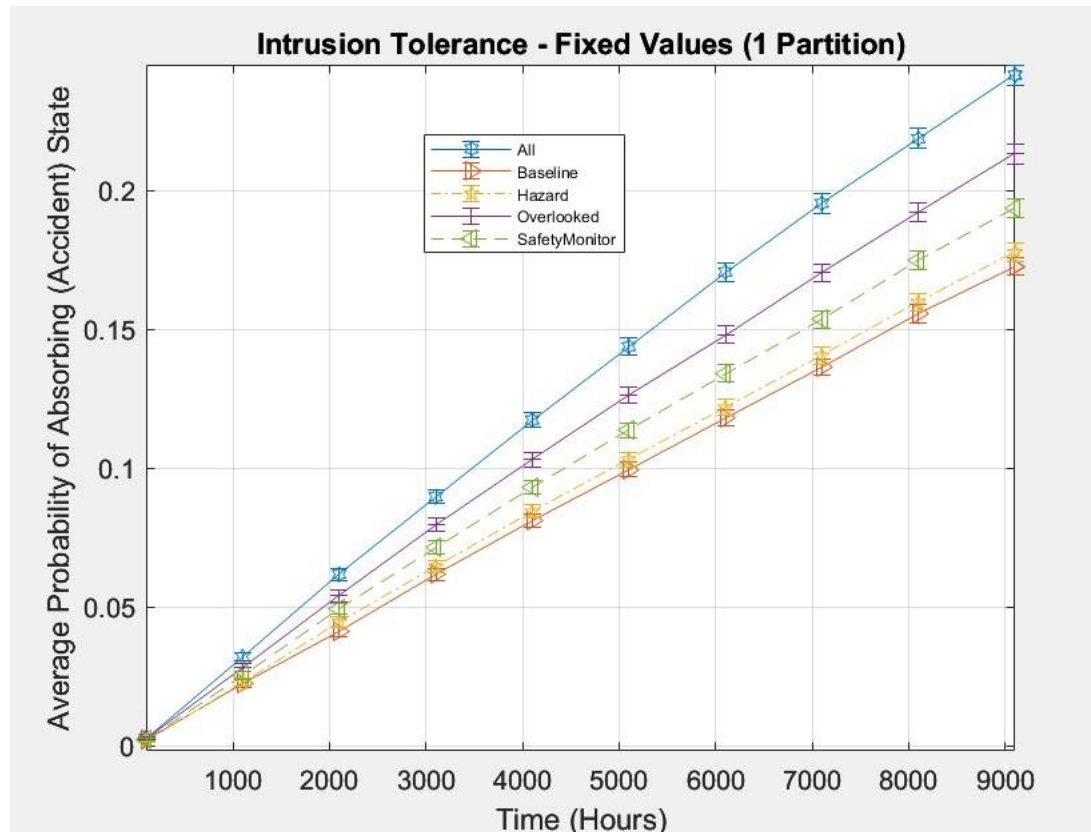


Figure 32: Intrusion Tolerance - Static Model (1 Partition)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
All	0.0028	0.0320	0.0618	0.0897	0.1174	0.1438	0.1707	0.1954	0.2187	0.2416
Baseline	0.0022	0.0225	0.0411	0.0617	0.0812	0.0995	0.1181	0.1365	0.1557	0.1727
Hazard	0.0020	0.0229	0.0447	0.0647	0.0843	0.1032	0.1218	0.1406	0.1599	0.1779
Overlooked	0.0026	0.0283	0.0540	0.0797	0.1031	0.1263	0.1482	0.1704	0.1923	0.2132
SafetyMonitor	0.0024	0.0255	0.0492	0.0716	0.0933	0.1137	0.1342	0.1537	0.1750	0.1938

Table 76: Intrusion Tolerance - Static (Raw Data - 1 Partition)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
All	4.6482e-04	0.0015	0.0021	0.0025	0.0028	0.0031	0.0033	0.0035	0.0036	0.0038
Baseline	4.0694e-04	0.0013	0.0017	0.0021	0.0024	0.0026	0.0028	0.0030	0.0032	0.0033
Hazard	3.8768e-04	0.0013	0.0018	0.0022	0.0024	0.0027	0.0029	0.0030	0.0032	0.0034
Overlooked	4.4808e-04	0.0015	0.0020	0.0024	0.0027	0.0029	0.0031	0.0033	0.0035	0.0036
SafetyMonitor	4.2890e-04	0.0014	0.0019	0.0023	0.0025	0.0028	0.0030	0.0032	0.0033	0.0035

Table 77: Intrusion Tolerance - Static (Confidence Levels - 1 Partition)

Looking at the results (Figure 32, Table 76, Table 77), it is seen that the risk of accidents in descending order is as follows:

1. **All:** The compromising of all the sections of the vehicle architecture.

2. **Overlooked:** The overlooking of hazards which are not immediately detected.
3. **SafetyMonitor:** The system enforcing the safety policies, on which attacks will affect its integrity.
4. **Hazard:** The immediate correct perception of real hazards.
5. **Baseline:** The basic safety level of the system when no adversary is present.

This is very similar to the other partitioned model, where the overlooking of hazards reduces the window of opportunity to mitigate them resulting in a greater risk to safety. The integrity of the safety monitor is also shown to be important, but the greatest risk is from compromising all parts of the architecture at once, suggesting that the next stage is to explore combination attacks on the architecture.

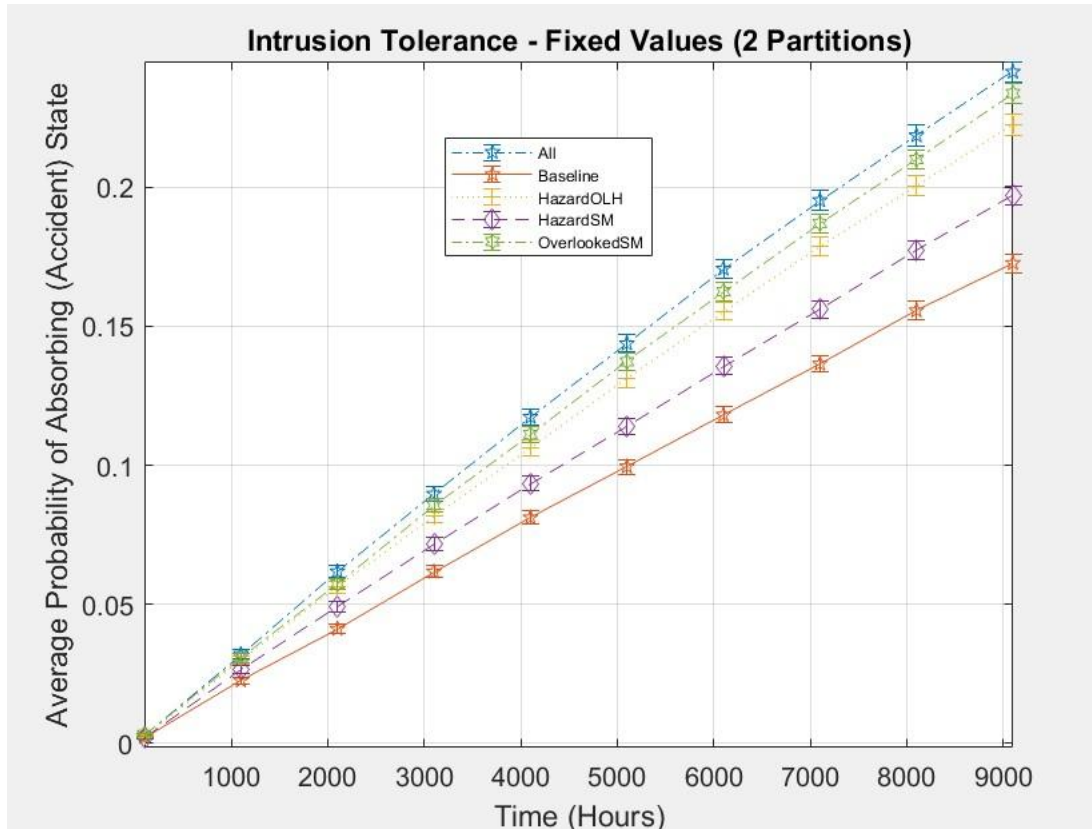


Figure 33: Intrusion Tolerance - Static Model (2 Partitions)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
All	0.0028	0.0320	0.0618	0.0897	0.1174	0.1438	0.1707	0.1954	0.2187	0.2416
Baseline	0.0022	0.0225	0.0411	0.0617	0.0812	0.0995	0.1181	0.1365	0.1557	0.1727
HazardOLH	0.0026	0.0300	0.0561	0.0818	0.1063	0.1311	0.1553	0.1788	0.2006	0.2225
HazardSM	0.0022	0.0265	0.0492	0.0717	0.0935	0.1139	0.1358	0.1561	0.1772	0.1970
OverlookedSM	0.0032	0.0306	0.0573	0.0855	0.1111	0.1373	0.1625	0.1870	0.2100	0.2338

Table 78: Intrusion Tolerance - Static (Raw Data - 2 Partitions)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
All	4.6482e-04	0.0015	0.0021	0.0025	0.0028	0.0031	0.0033	0.0035	0.0036	0.0038
Baseline	4.0694e-04	0.0013	0.0017	0.0021	0.0024	0.0026	0.0028	0.0030	0.0032	0.0033
HazardOLH	4.4978e-04	0.0015	0.0020	0.0024	0.0027	0.0030	0.0032	0.0034	0.0035	0.0036
HazardSM	4.0694e-04	0.0014	0.0019	0.0023	0.0026	0.0028	0.0030	0.0032	0.0033	0.0035
OverlookedSM	4.9351e-04	0.0015	0.0020	0.0025	0.0028	0.0030	0.0032	0.0034	0.0036	0.0037

Table 79: Intrusion Tolerance - Static (Confidence Levels - 2 Partitions)

For the combined partitions, the results shown (Figure 33, Table 78, Table 79) indicate the following risk from highest to lowest when it comes to safety:

1. **All:** Compromising of all three partitions.
2. **OverlookedSM (Overlooked + Safety Monitor):** Overlooked hazards and the safety monitor integrity being compromised at the same time, meaning hazards may not be detected at all and the safety monitor is less effective at enforcing policies.
3. **HazardOLH (Hazard + Overlooked):** Hazard perception with overlooked hazards.
4. **HazardSM (Hazard + Safety Monitor):** Hazard perception and safety monitor integrity.
5. **Baseline:** Safety of the system without attacks present.

From this combination of results, the ‘Overlooked’ component continues to be serious, and when combined with the safety monitor integrity being compromised, this becomes even more serious, presenting the highest risk that does not involve compromising all the components at once. Results shown above indicate significant benefits of intrusion tolerance, with results still being worse than the baseline but significantly better than if intrusion tolerance was not present.

6.2.4.1 Discussion

The results from attacking partitions in the presence of static intrusion tolerance indicate the following:

- Attacks on the late/overlooked hazard perception component of the vehicle have the most individual effect. This is because the vehicle will miss its fair share of hazards on time, especially in a constantly changing environment so if the unit designed to compensate for this is not working properly then the risk to safety will be very high. For intrusion tolerance, this translates into a need for more replicas and higher cleansing rates to truly eliminate most of the risk here.
- Compromising the safety monitor with the late/overlooked hazard system at the same time is the most dangerous combination of two components being compromised. By having a safety monitor with poor integrity at the same time as failing to detect missed hazards later means the vehicle is now missing hazards while not behaving safely, which can lead to a lot more accidents. By applying stronger intrusion tolerance to both components at a high level, it is possible to lower the risk to safety.
- Compromising all components of the vehicle together is still the worst-case scenario here, as doing this means that not only is the perception not working properly, but the safety monitor cannot attempt to enforce safety for the vehicle either. While it is tempting to apply it to all the components, this risks a lot of overhead, so by applying

it to perception with greater strength this can be overcome. Another option would be to try system-wide intrusion tolerance if it can be done efficiently.

- Compared to the regular partitioned model and defence-in-depth, intrusion tolerance for this fixed model is in fact found to be slightly more effective against an adversary, giving lower risks to safety.
- Results indicate significant benefits of intrusion tolerance, with results still being worse than the baseline but significantly better than if intrusion tolerance was not present.
- There is only a single absorbing state, so the results are limited to one type of accident, when there could be many different scenarios where a safety failure is not necessarily catastrophic, limiting the extent to which the model can assess safety.
- A limitation of the model is that it assumes adversaries that occur at a fixed rate with some success probability, which is typically not 100% meaning that the model does not account for an adversary that can perfectly execute an attack every time without issue (i.e. I assume a limited level of expertise from the adversary).
- It must also be noted that while the model is limited to Exponentially distributed timed activities only, this is not a major issue as according to the findings of (Littlewood, 1979), all Markov models of this type are asymptotically Exponential regardless of distributions meaning that my results for the risk would give an Exponential result over time anyway.

6.2.5 Dynamic Model (Replicas)

Going forward from the static model, an additional stage is now added, replacing the number of replicas with a nonzero value indicating the number of levels of replicas which can be increased or decreased from (i.e. going from 4 to 7 to 10 replicas depending on if the system is being compromised for a lower number of replicas). This model is an attempt to adapt the CAV response to increasing or decreasing adversary intensity and competence, with this initial model only being applied to a fixed adversary to begin with (i.e. no change in attack parameters yet).

In the table below the parameters are shown, where the attack model is still fixed, with an upper bound for replicas (e.g. `sojmaxReps`) because of the use of an array for the replicas which needs to have its size initially declared. The models were simulated across multiple points in time from 100-9100 hours in 1000-hour steps, using Monte Carlo simulations for 50000 batches in steps of 1000 batches to observe the change in accident probability over the period for which the CAV is observed.

Parameter	Value(s)
AccidentScale	10.0
CH2AccRate	1.0E-4
CH2OKRate	856.3
DelayScale	1000.0
FH2AccRate	1.0E-5
FH2OKRate	856.3
HazardRate	197.4
HazardScale	10.0
LateCH2AccScale	10.0
MH2AccRate	0.001
MH2CHRate	2250.0
OH2OKRate	1000.0
OH2CH scale	10.0
percCleanse	0.3
percmaxReps	100
percLevels	3
percRate	0.1
percSuccess	0.2
sojCleanse	0.3
sojmaxReps	100
sojLevels	3
sojRate	0.1
sojSuccess	0.2
smCleanse	0.3
safemaxReps	100
safeLevels	3
smRate	0.1
smSuccess	0.2
percEnable	[0, 1]
sojEnable	[0, 1]
smEnable	[0, 1]
falseHazardProb	1.0E-4
isRSSperfect	0
missHazardProb	0.05

Table 80: Intrusion Tolerance Parameters - Dynamic Replicas

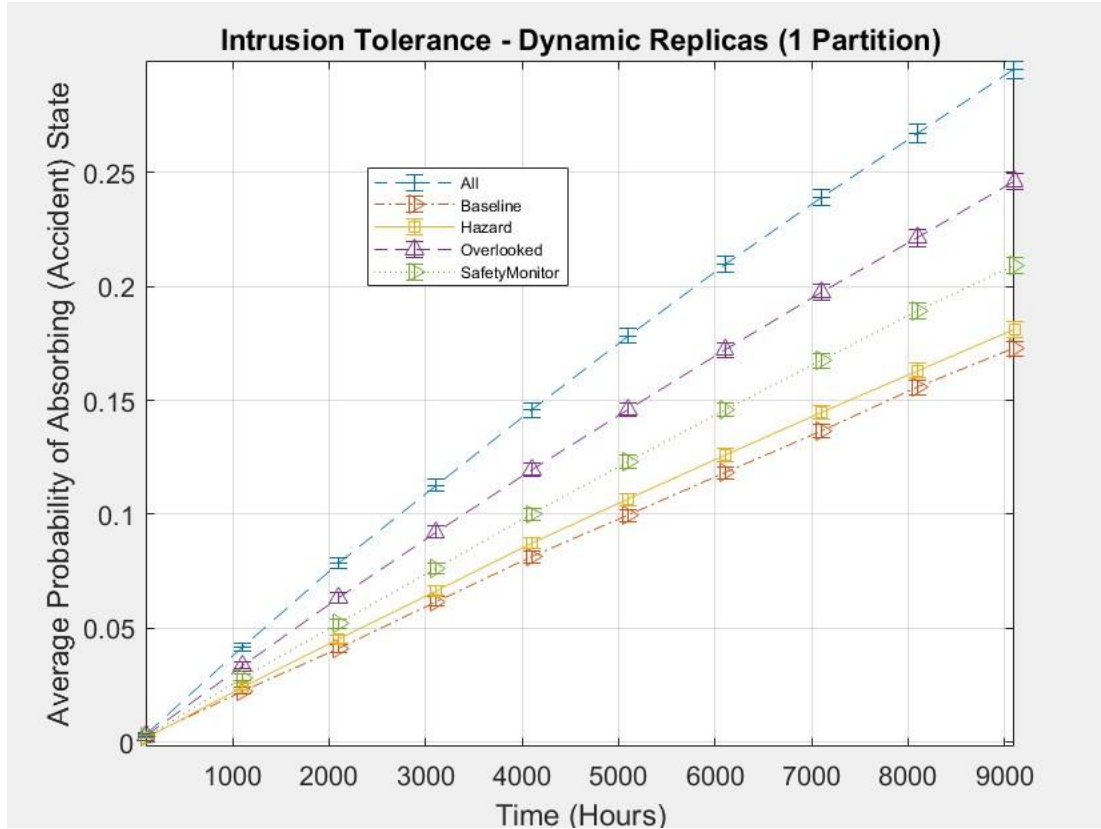


Figure 34: Intrusion Tolerance - Dynamic Replicas (1 Partition)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
All	0.0038	0.0416	0.0785	0.1128	0.1457	0.1783	0.2099	0.2390	0.2672	0.2951
Baseline	0.0022	0.0225	0.0411	0.0617	0.0812	0.0995	0.1181	0.1365	0.1557	0.1727
Hazard	0.0020	0.0242	0.0453	0.0664	0.0873	0.1065	0.1258	0.1447	0.1630	0.1812
Overlooked	0.0027	0.0336	0.0633	0.0922	0.1197	0.1460	0.1720	0.1974	0.2213	0.2461
SafetyMonitor	0.0023	0.0282	0.0520	0.0760	0.0999	0.1231	0.1460	0.1674	0.1891	0.2090

Table 81: Intrusion Tolerance - Dynamic Replicas (Raw Data - 1 Partition)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
All	5.3648e-04	0.0017	0.0024	0.0028	0.0031	0.0034	0.0036	0.0037	0.0039	0.0040
Baseline	4.0694e-04	0.0013	0.0017	0.0021	0.0024	0.0026	0.0028	0.0030	0.0032	0.0033
Hazard	3.8768e-04	0.0013	0.0018	0.0022	0.0025	0.0027	0.0029	0.0031	0.0032	0.0034
Overlooked	4.5485e-04	0.0016	0.0021	0.0025	0.0028	0.0031	0.0033	0.0035	0.0036	0.0038
SafetyMonitor	4.2171e-04	0.0015	0.0019	0.0023	0.0026	0.0029	0.0031	0.0033	0.0034	0.0036

Table 82: Intrusion Tolerance - Dynamic Replicas (Confidence Levels - 1 Partition)

Looking at the results (Figure 34, Table 81, Table 82), it is seen that the risk of accidents in descending order is as follows:

1. **All:** The compromising of all the sections of the vehicle architecture.
2. **Overlooked:** The overlooking of hazards which are not immediately detected.
3. **SafetyMonitor:** The system enforcing the safety policies, on which attacks will affect its integrity.
4. **Hazard:** The immediate correct perception of real hazards.
5. **Baseline:** The basic safety level of the system when no adversary is present.

This is very similar to the other partitioned model, where the overlooking of hazards reduces the window of opportunity to mitigate them resulting in a greater risk to safety. The integrity of the safety monitor is also shown to be important, but the greatest risk is from compromising all parts of the architecture at once, suggesting that the next stage is to explore combination attacks on the architecture.

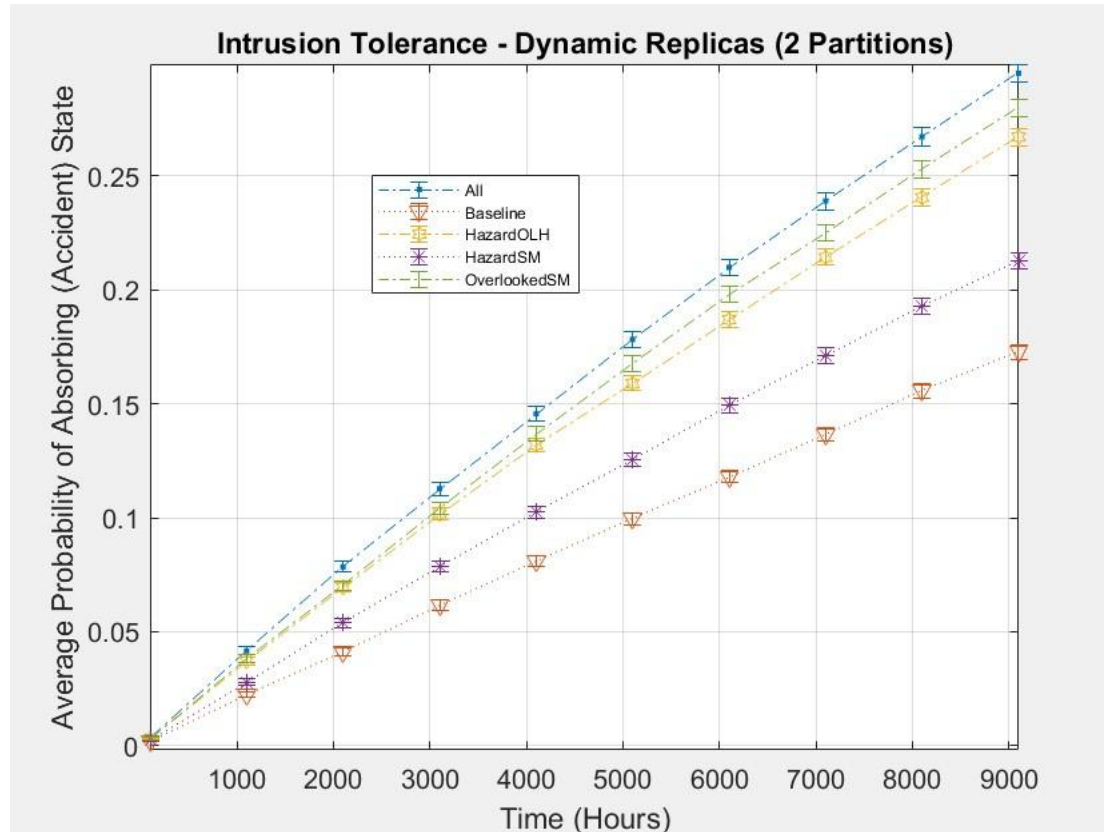


Figure 35: Intrusion Tolerance - Dynamic Replicas (2 Partitions)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
All	0.0038	0.0416	0.0785	0.1128	0.1457	0.1783	0.2099	0.2390	0.2672	0.2951
Baseline	0.0022	0.0225	0.0411	0.0617	0.0812	0.0995	0.1181	0.1365	0.1557	0.1727
HazardOLH	0.0029	0.0372	0.0695	0.1018	0.1319	0.1590	0.1870	0.2143	0.2406	0.2671
HazardSM	0.0023	0.0279	0.0539	0.0785	0.1025	0.1257	0.1493	0.1711	0.1928	0.2128
OverlookedSM	0.0031	0.0382	0.0702	0.1042	0.1369	0.1677	0.1979	0.2251	0.2531	0.2798

Table 83: Intrusion Tolerance - Dynamic Replicas (Raw Data - 2 Partitions)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
All	5.3648e-04	0.0017	0.0024	0.0028	0.0031	0.0034	0.0036	0.0037	0.0039	0.0040
Baseline	4.0694e-04	0.0013	0.0017	0.0021	0.0024	0.0026	0.0028	0.0030	0.0032	0.0033
HazardOLH	4.7297e-04	0.0017	0.0022	0.0027	0.0030	0.0032	0.0034	0.0036	0.0037	0.0039
HazardSM	4.1807e-04	0.0014	0.0020	0.0024	0.0027	0.0029	0.0031	0.0033	0.0025	0.0036
OverlookedSM	4.4885e-04	0.0017	0.0022	0.0027	0.0030	0.0033	0.0035	0.0037	0.0038	0.0039

Table 84: Intrusion Tolerance - Dynamic Replicas (Confidence Levels - 2 Partitions)

For the combined partitions, the results shown (Figure 35, Table 83, Table 84) indicate the following risk from highest to lowest when it comes to safety:

1. **All:** Compromising of all three partitions.

2. **OverlookedSM (Overlooked + Safety Monitor):** Overlooked hazards and the safety monitor integrity being compromised at the same time, meaning hazards may not be detected at all and the safety monitor is less effective at enforcing policies.
3. **HazardOLH (Hazard + Overlooked):** Hazard perception with overlooked hazards.
4. **HazardSM (Hazard + Safety Monitor):** Hazard perception and safety monitor integrity.
5. **Baseline:** Safety of the system without attacks present.

From this combination of results, the ‘Overlooked’ component continues to be serious, and when combined with the safety monitor integrity being compromised, this becomes even more serious, presenting the highest risk that does not involve compromising all the components at once.

6.2.5.1 Discussion

The results from attacking partitions with dynamic intrusion tolerance indicate the following:

- Attacks on the late/overlooked hazard perception component of the vehicle have the most individual effect. This is because the vehicle will miss its fair share of hazards on time, especially in a constantly changing environment so if the unit designed to compensate for this is not working properly then the risk to safety will be very high. This suggests a need for stronger measures (e.g. high cleansing rates) to be applied to the replicas at these components.
- Compromising the hazard perception with the late/overlooked hazard system at the same time is the most dangerous combination of two components being compromised. This is because the detection rate of hazards deteriorates while the ones not immediately detected are also not being detected as often, so the perception is heavily weakened meaning that the CAV is liable to have a lot more accidents. By strengthening the intrusion tolerance further or improving the dynamic approach, this can still be mitigated.
- Compromising all components of the vehicle together is still the worst-case scenario here, as doing this means that not only is the perception not working properly, but the safety monitor cannot attempt to enforce safety for the vehicle either.
- The dynamic model has been less effective than the fixed model in this case, with defence-in-depth being a superior option compared to this dynamic replication model.
- There is only a single absorbing state, so the results are limited to one type of accident, when there could be many different scenarios where a safety failure is not necessarily catastrophic, limiting the extent to which the model can assess safety.
- A limitation of the model is that it assumes adversaries that occur at a fixed rate with some success probability, which is typically not 100% meaning that the model does not account for an adversary that can perfectly execute an attack every time without issue (i.e. I assume a limited level of expertise from the adversary).
- It must also be noted that while the model is limited to Exponentially distributed timed activities only, this is not a major issue as according to the findings of (Littlewood, 1979), all Markov models of this type are asymptotically Exponential irrespective of the distributions used meaning that my results for the risk would give an Exponential result over time anyway.

6.2.6 Dynamic Model (Replicas & Attacks)

So far, the models have just been varying the number of replicas in the dynamic stage. In this one, the attack model is adjusted so that it varies every 1000 hours for the attack rate between values of 0.01, 0.05, 0.1 and 0.3, going in a cycle throughout the model simulations. The table below shows fixed cleansing and the upper bound for the replicas, so this model now has a more dynamic and unpredictable adversary at the same time as an adaptive change to the number of replicas in the presence of the adversary. The purpose of this model is to adapt to an adversary of varying intensity and competence once again but in this case the adversary now has dynamic behaviour making them significantly less predictable.

The models were simulated across multiple points in time from 100-9100 hours in 1000-hour steps, using Monte Carlo simulations for 50000 batches in steps of 1000 batches to observe the change in accident probability over the period for which the CAV is observed.

Parameter	Value(s)
AccidentScale	10.0
CH2AccRate	1.0E-4
CH2OKRate	856.3
DelayScale	1000.0
FH2AccRate	1.0E-5
FH2OKRate	856.3
HazardRate	197.4
HazardScale	10.0
LateCH2AccScale	10.0
MH2AccRate	0.001
MH2CHRate	2250.0
OH2OKRate	1000.0
OH2CH scale	10.0
dynRate	0.001
percCleanse	0.3
percmaxReps	100
percLevels	3
percSuccess	0.2
sojCleanse	0.3
sojmaxReps	100
sojLevels	3
sojSuccess	0.2
smCleanse	0.3
safemaxReps	100
safeLevels	3
smSuccess	0.2
percEnable	[0, 1]
sojEnable	[0, 1]
smEnable	[0, 1]
falseHazardProb	1.0E-4
isRSSperfect	0
missHazardProb	0.05

Table 85: Intrusion Tolerance Parameters - Dynamic Replicas & Adversary

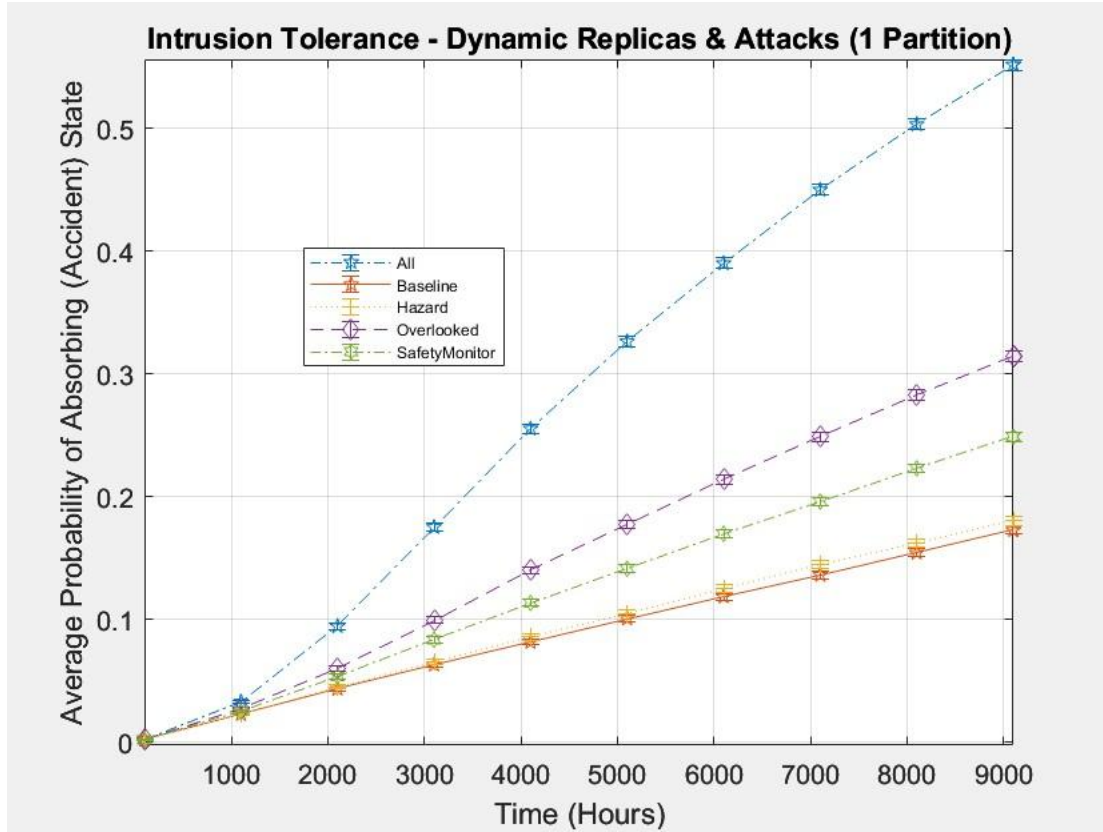


Figure 36: Intrusion Tolerance - Dynamic Replicas & Attacks (1 Partition)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
All	0.0021	0.0330	0.0945	0.1752	0.2553	0.3263	0.3900	0.4497	0.5031	0.5516
Baseline	0.0020	0.0230	0.0434	0.0632	0.0816	0.1006	0.1186	0.1363	0.1543	0.1728
Hazard	0.0021	0.0233	0.0452	0.0659	0.0857	0.1047	0.1252	0.1449	0.1622	0.1802
Overlooked	0.0020	0.0273	0.0605	0.0995	0.1399	0.1773	0.2138	0.2489	0.2826	0.3144
SafetyMonitor	0.0022	0.0251	0.0536	0.0835	0.1134	0.1413	0.1698	0.1961	0.2230	0.2488

Table 86: Intrusion Tolerance - Dynamic Replicas & Attacks (Raw Data – 1 Partition)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
All	4.0506e-04	0.0016	0.0026	0.0033	0.0038	0.0041	0.0043	0.0044	0.0044	0.0044
Baseline	3.9356e-04	0.0013	0.0018	0.0021	0.0024	0.0026	0.0028	0.0030	0.0032	0.0033
Hazard	4.0316e-04	0.0013	0.0018	0.0022	0.0025	0.0027	0.0029	0.0031	0.0032	0.0034
Overlooked	3.9550e-04	0.0014	0.0021	0.0026	0.0030	0.0033	0.0036	0.0038	0.0039	0.0041
SafetyMonitor	4.0882e-04	0.0014	0.0020	0.0024	0.0028	0.0031	0.0033	0.0035	0.0036	0.0038

Table 87: Intrusion Tolerance - Dynamic Replicas & Attacks (Confidence Levels - 1 Partition)

Looking at the results (Figure 36, Table 86, Table 87), it is seen that the risk of accidents in descending order is as follows:

1. **All:** The compromising of all the sections of the vehicle architecture.
2. **Overlooked:** The overlooking of hazards which are not immediately detected.
3. **SafetyMonitor:** The system enforcing the safety policies, on which attacks will affect its integrity.
4. **Hazard:** The immediate correct perception of real hazards.
5. **Baseline:** The basic safety level of the system when no adversary is present.

This is very similar to the other partitioned model, where the overlooking of hazards reduces the window of opportunity to mitigate them resulting in a greater risk to safety. The integrity of the safety monitor is also shown to be important, but the greatest risk is from compromising all parts of the architecture at once, suggesting that the next stage is to explore combination attacks on the architecture.

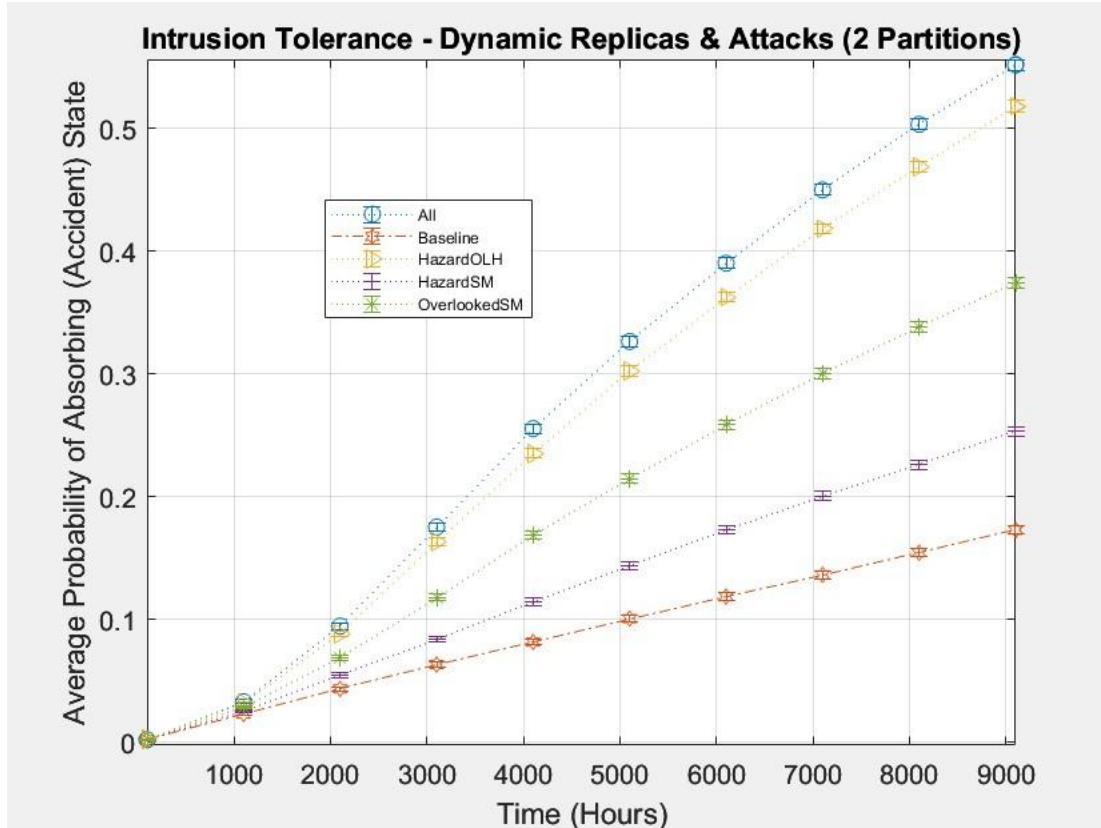


Figure 37: Intrusion Tolerance - Dynamic Replicas & Attacks (2 Partitions)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
All	0.0021	0.0330	0.0945	0.1752	0.2553	0.3263	0.3900	0.4497	0.5031	0.5516
Baseline	0.0020	0.0230	0.0434	0.0632	0.0816	0.1006	0.1186	0.1363	0.1543	0.1728
HazardOLH	0.0022	0.0316	0.0884	0.1629	0.2352	0.3024	0.3625	0.4180	0.4688	0.5179
HazardSM	0.0020	0.0255	0.0547	0.0842	0.1144	0.1434	0.1727	0.2007	0.2261	0.2532
OverlookedSM	0.0019	0.0284	0.0690	0.1181	0.1688	0.2147	0.2585	0.2999	0.3384	0.3738

Table 88: Intrusion Tolerance - Dynamic Replicas & Attacks (Raw Data - 2 Partitions)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
All	4.0506e-04	0.0016	0.0026	0.0033	0.0038	0.0041	0.0043	0.0044	0.0044	0.0044
Baseline	3.9356e-04	0.0013	0.0018	0.0021	0.0024	0.0026	0.0028	0.0030	0.0032	0.0033
HazardOLH	4.1439e-04	0.0015	0.0025	0.0032	0.0037	0.0040	0.0042	0.0043	0.0044	0.0044
HazardSM	3.9161e-04	0.0014	0.0020	0.0024	0.0028	0.0031	0.0033	0.0035	0.0037	0.0038
OverlookedSM	3.8371e-04	0.0015	0.0022	0.0028	0.0033	0.0036	0.0038	0.0040	0.0041	0.0042

Table 89: Intrusion Tolerance - Dynamic Replicas & Attacks (Confidence Levels - 2 Partitions)

For the combined partitions, the results shown (Figure 37, Table 88, Table 89) indicate the following risk from highest to lowest when it comes to safety:

1. **All:** Compromising of all three partitions.
2. **HazardOLH (Hazard + Overlooked):** Hazard perception with overlooked hazards.
3. **OverlookedSM (Overlooked + Safety Monitor):** Overlooked hazards and the safety monitor integrity being compromised at the same time, meaning hazards may not be detected at all and the safety monitor is less effective at enforcing policies.
4. **HazardSM (Hazard + Safety Monitor):** Hazard perception and safety monitor integrity.
5. **Baseline:** Safety of the system without attacks present.

From this combination of results, the ‘Overlooked’ component continues to be an issue. However, the hazard perception combined with overlooking hazards is the most serious one when dynamic attacks are introduced at the same time as dynamic replicas.

6.2.6.1 Discussion

The results from attacking partitions with dynamic intrusion tolerance and attacks indicate the following:

- Attacks on the late/overlooked hazard perception component of the vehicle have the most individual effect. This is because the vehicle will miss its fair share of hazards on time, especially in a constantly changing environment so if the unit designed to compensate for this is not working properly then the risk to safety will be very high. The presence of a dynamic adversary means that intrusion tolerance measures must become even more robust, with enough strength to cover a range of adversaries, by increasing the cleansing rates, even giving them a dynamic property rather than just with replication.
- Compromising the hazard perception with the late/overlooked hazard system at the same time is the most dangerous combination of two components being compromised. This is because the detection rate of hazards deteriorates while the ones not immediately detected are also not being detected as often, so the perception is heavily weakened meaning that the CAV is liable to have a lot more accidents. This is a significant risk now that the behaviour of the adversary is no longer predictable, so both the replication and cleansing rates should be dynamic to limit the window of time such an adversary has.
- Compromising all components of the vehicle together is still the worst-case scenario here, as doing this means that not only is the perception not working properly, but the safety monitor cannot attempt to enforce safety for the vehicle either.
- For this model, where varied attack intensity is introduced, the dynamic replication was inadequate for intrusion tolerance to be effective, as the adversary now had increasing intensity while the intrusion tolerance was at the same cleansing rate, so the probability of overcoming the replicas was significantly higher.
- There is only a single absorbing state, so the results are limited to one type of accident, when there could be many different scenarios where a safety failure is not necessarily catastrophic, limiting the extent to which the model can assess safety.
- A limitation of the model is that it assumes adversaries that occur at a fixed rate with some success probability, which is typically not 100% meaning that the model does not account for an adversary that can perfectly execute an attack every time without issue (i.e. I assume a limited level of expertise from the adversary).

- It must also be noted that while the model is limited to Exponentially distributed timed activities only, this is not a major issue as according to the findings of (Littlewood, 1979), all Markov models of this type are asymptotically Exponential no matter what distributions are used meaning that my results for the risk would give an Exponential result over time anyway.

6.2.7 Dynamic Model (Replicas, Attacks & Cleansing)

This model, looking at the previous results, aims to focus on now introducing dynamic cleansing which alternates between values of 0.3, 0.5, 1.0 and 2.0 in a cycle to compensate for the effects of the increasingly strong adversary, while working at the same rate of 1 in 1000 hours, which is the same as the adversary, as seen in the table below. Based off the prior dynamic models the purpose here is similar, with a dynamic adversary and replicas but also introducing dynamic cleansing to attempt to offset the effect of the adversary.

The models were simulated across multiple points in time from 100-9100 hours in 1000-hour steps, using Monte Carlo simulations for 50000 batches in steps of 1000 batches to observe the change in accident probability over the period for which the CAV is observed.

Parameter	Value(s)
AccidentScale	10.0
CH2AccRate	1.0E-4
CH2OKRate	856.3
DelayScale	1000.0
FH2AccRate	1.0E-5
FH2OKRate	856.3
HazardRate	197.4
HazardScale	10.0
LateCH2AccScale	10.0
MH2AccRate	0.001
MH2CHRate	2250.0
OH2OKRate	1000.0
OH2CH scale	10.0
dynRate	0.001
dynPerc	0.001
dynSoj	0.001
dynSafe	0.001
percmaxReps	100
percLevels	3
percSuccess	0.2
sojmaxReps	100
sojLevels	3
sojSuccess	0.2
safemaxReps	100
safeLevels	3
smSuccess	0.2
percEnable	[0, 1]
sojEnable	[0, 1]
smEnable	[0, 1]
falseHazardProb	1.0E-4
isRSSperfect	0
missHazardProb	0.05

Table 90: Intrusion Tolerance Parameters - Dynamic Replicas, Adversary and Cleansing

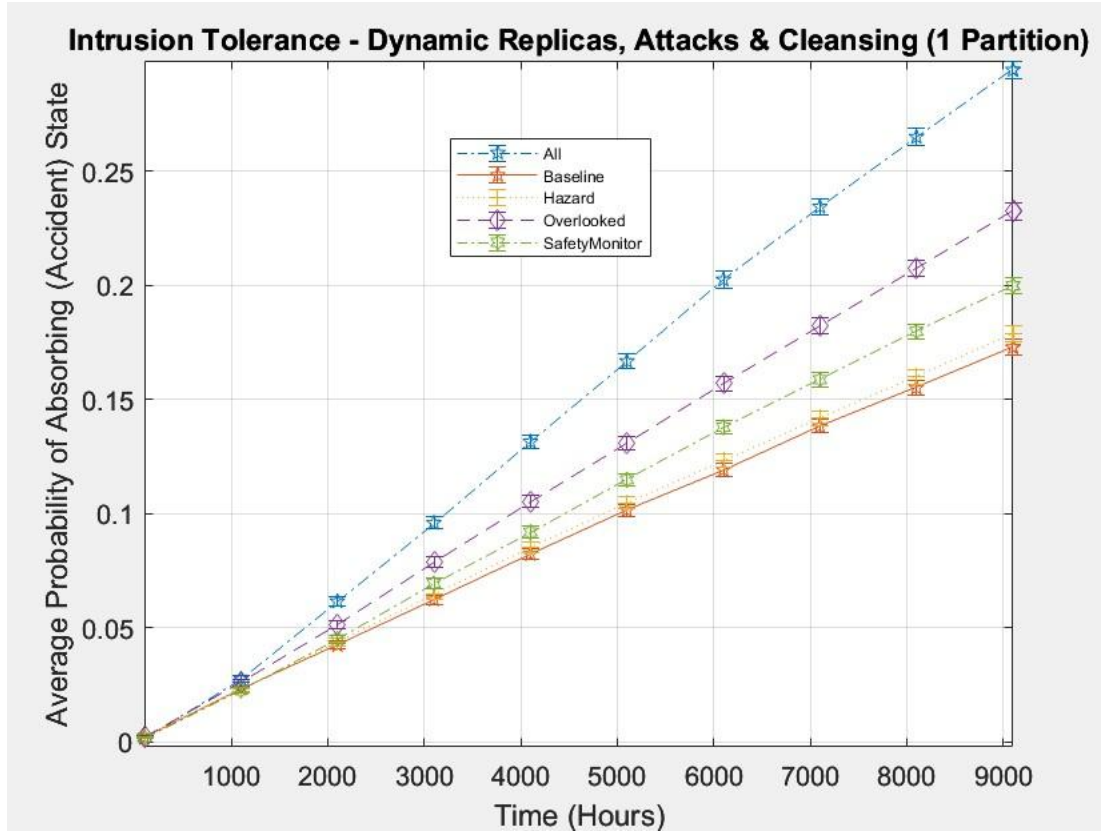


Figure 38: Intrusion Tolerance - Dynamic Replicas, Attacks & Cleansing (1 Partition)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
All	0.0018	0.0276	0.0615	0.0959	0.1315	0.1667	0.2024	0.2343	0.2649	0.2943
Baseline	0.0021	0.0232	0.0424	0.0623	0.0823	0.1015	0.1192	0.1382	0.1551	0.1728
Hazard	0.0021	0.0232	0.0439	0.0648	0.0854	0.1048	0.1231	0.1417	0.1600	0.1787
Overlooked	0.0023	0.0260	0.0514	0.0788	0.1053	0.1311	0.1570	0.1822	0.2072	0.2324
SafetyMonitor	0.0019	0.0226	0.0451	0.0694	0.0919	0.1148	0.1378	0.1586	0.1798	0.1997

Table 91: Intrusion Tolerance - Dynamic Replicas, Attacks & Cleansing (Raw Data - 1 Partition)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
All	3.7565e-04	0.0014	0.0021	0.0026	0.0030	0.0033	0.0035	0.0037	0.0039	0.0040
Baseline	4.0126e-04	0.0013	0.0018	0.0021	0.0024	0.0026	0.0028	0.0030	0.0032	0.0033
Hazard	3.9743e-04	0.0013	0.0018	0.0022	0.0024	0.0027	0.0029	0.0031	0.0032	0.0034
Overlooked	4.2171e-04	0.0014	0.0019	0.0024	0.0027	0.0030	0.0032	0.0034	0.0036	0.0037
SafetyMonitor	3.8371e-04	0.0013	0.0018	0.0022	0.0025	0.0028	0.0030	0.0032	0.0034	0.0035

Table 92: Intrusion Tolerance - Dynamic Replicas, Attacks & Cleansing (Confidence Levels - 1 Partition)

Looking at the results (Figure 38, Table 91, Table 92), it is seen that the risk of accidents in descending order is as follows:

1. **All:** The compromising of all the sections of the vehicle architecture.
2. **Overlooked:** The overlooking of hazards which are not immediately detected.
3. **SafetyMonitor:** The system enforcing the safety policies, on which attacks will affect its integrity.
4. **Hazard:** The immediate correct perception of real hazards.

5. **Baseline:** The basic safety level of the system when no adversary is present.

This is very similar to the other partitioned model, where the overlooking of hazards reduces the window of opportunity to mitigate them resulting in a greater risk to safety. The integrity of the safety monitor is also shown to be important, but the greatest risk is from compromising all parts of the architecture at once, suggesting that the next stage is to explore combination attacks on the architecture.

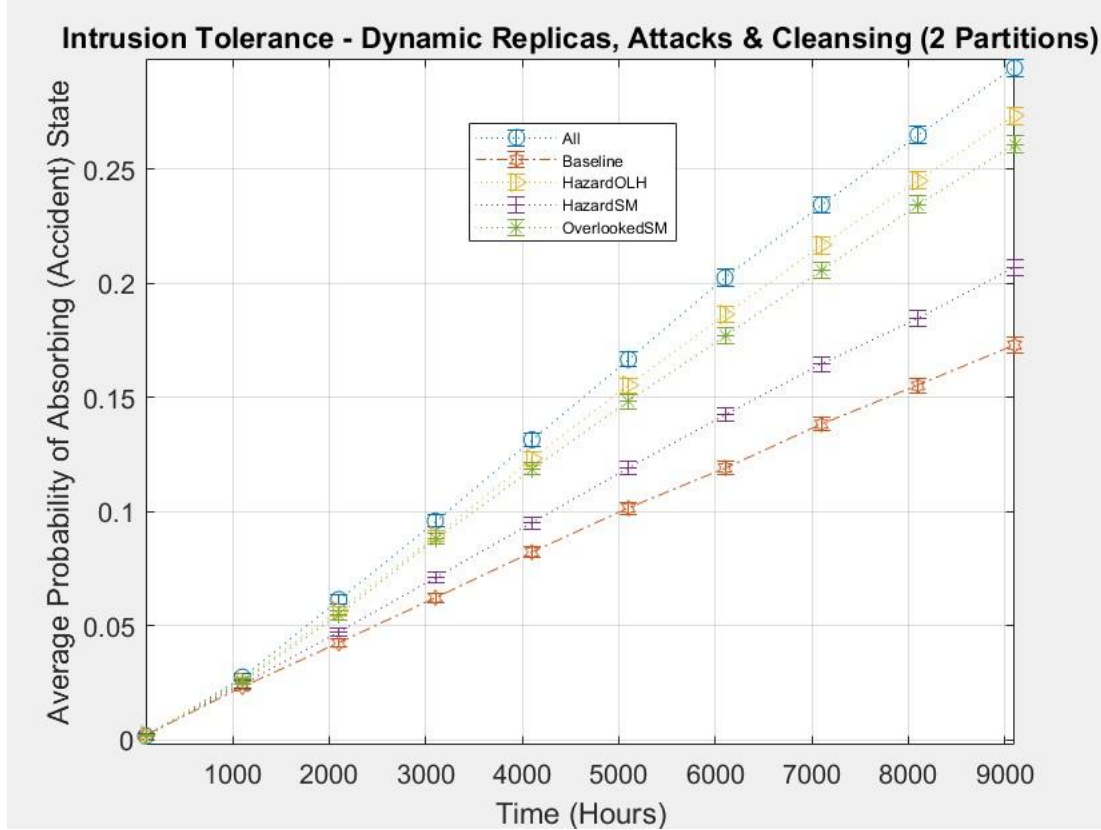


Figure 39: Intrusion Tolerance - Dynamic Replicas, Attacks & Cleansing (2 Partitions)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
All	0.0018	0.0276	0.0615	0.0959	0.1315	0.1667	0.2024	0.2324	0.2649	0.2943
Baseline	0.0021	0.0232	0.0424	0.0623	0.0823	0.1015	0.1192	0.1382	0.1551	0.1728
HazardOLH	0.0021	0.0259	0.0565	0.0892	0.1231	0.1552	0.1863	0.2165	0.2449	0.2732
HazardSM	0.0023	0.0243	0.0472	0.0714	0.0949	0.1192	0.1424	0.1646	0.1847	0.2066
OverlookedSM	0.0019	0.0256	0.0548	0.0883	0.1188	0.1482	0.1771	0.2058	0.2344	0.2608

Table 93: Intrusion Tolerance - Dynamic Replicas, Attacks & Cleansing (Raw Data - 2 Partitions)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
All	3.7565e-04	0.0014	0.0021	0.0026	0.0030	0.0033	0.0035	0.0037	0.0039	0.0040
Baseline	4.0126e-04	0.0013	0.0018	0.0021	0.0024	0.0026	0.0028	0.0030	0.0032	0.0033
HazardOLH	4.0316e-04	0.0014	0.0020	0.0025	0.0029	0.0032	0.0034	0.0036	0.0038	0.0039
HazardSM	4.1623e-04	0.0014	0.0019	0.0023	0.0026	0.0028	0.0031	0.0033	0.0034	0.0035
OverlookedSM	3.7768e-04	0.0014	0.0020	0.0025	0.0028	0.0031	0.0033	0.0035	0.0037	0.0038

Table 94: Intrusion Tolerance - Dynamic Replicas, Attacks & Cleansing (Confidence Levels - 2 Partitions)

For the combined partitions, the results shown (Figure 39, Table 93, Table 94) indicate the following risk from highest to lowest when it comes to safety:

1. **All:** Compromising of all three partitions.
2. **HazardOLH (Hazard + Overlooked):** Hazard perception with overlooked hazards.
3. **OverlookedSM (Overlooked + Safety Monitor):** Overlooked hazards and the safety monitor integrity being compromised at the same time, meaning hazards may not be detected at all and the safety monitor is less effective at enforcing policies.
4. **HazardSM (Hazard + Safety Monitor):** Hazard perception and safety monitor integrity.
5. **Baseline:** Safety of the system without attacks present.

From this combination of results, the ‘Overlooked’ component continues to be an issue. However, the hazard perception combined with overlooking hazards is the most serious one when dynamic attacks and cleansing are introduced at the same time as dynamic replicas.

6.2.7.1 Discussion

The results from attacking partitions with dynamic intrusion tolerance and attacks indicate the following:

- Attacks on the late/overlooked hazard perception component of the vehicle have the most individual effect. This is because the vehicle will miss its fair share of hazards on time, especially in a constantly changing environment so if the unit designed to compensate for this is not working properly then the risk to safety will be very high. While the dynamic cleansing does reduce the risk, there may be a need for a different approach on determining when the system should adapt.
- Compromising the hazard perception with the late/overlooked hazard system at the same time is the most dangerous combination of two components being compromised. This is because the detection rate of hazards deteriorates while the ones not immediately detected are also not being detected as often, so the perception is heavily weakened meaning that the CAV is liable to have a lot more accidents. Therefore, perception modules will need higher cleansing rates of a dynamic nature applied to them to improve safety in the presence of an unpredictable adversary.
- Compromising all components of the vehicle together is still the worst-case scenario here, as doing this means that not only is the perception not working properly, but the safety monitor cannot attempt to enforce safety for the vehicle either. There is potential for a system-wide approach but there is the possibility of this having a lot of overhead so securing the perception may be sufficient to mitigate the risks.
- For this model, where varied attack intensity is combined with dynamic cleansing rate, the dynamic replication was inadequate for intrusion tolerance to be more effective, as the dynamic behaviour of the adversary was counteracted by having increasing rates of cleansing which could match.
- Despite being better than the model with dynamic attacks, the model was still less effective than dynamic replication on its own despite introducing cleansing and less effective than defence-in-depth.
- There is only a single absorbing state, so the results are limited to one type of accident, when there could be many different scenarios where a safety failure is not necessarily catastrophic, limiting the extent to which the model can assess safety.

- A limitation of the model is that it assumes adversaries that occur at a fixed rate with some success probability, which is typically not 100% meaning that the model does not account for an adversary that can perfectly execute an attack every time without issue (i.e. I assume a limited level of expertise from the adversary).
- It must also be noted that while the model is limited to Exponentially distributed timed activities only, this is not a major issue as according to the findings of (Littlewood, 1979), all Markov models of this type are asymptotically Exponential no matter the choice of distributions meaning that my results for the risk would give an Exponential result over time anyway.

6.2.8 Sensitivity Analysis

The choosing of the number of replicas was done in this area, by testing different cleansing rates and later between values of 4,7 and 10 replicas as shown in the table below. The values chosen were done so in a way that would allow the vulnerable areas in the presence of an attack on this new type of model to still be observable both numerically and graphically. The models were simulated across multiple points in time from 100-9100 hours in 1000-hour steps, using Monte Carlo simulations for 1000 batches in steps of 1000 batches to observe the change in accident probability over the period for which the CAV is observed.

Parameter	Value(s)
AccidentScale	10.0
CH2AccRate	1.0E-4
CH2OKRate	856.3
DelayScale	1000.0
FH2AccRate	1.0E-5
FH2OKRate	856.3
HazardRate	197.4
HazardScale	10.0
LateCH2AccScale	10.0
MH2AccRate	0.001
MH2CHRate	2250.0
OH2OKRate	1000.0
OH2CH scale	10.0
percCleanse	[0.001,0.01,0.05,0.1,0.3,0.5,1.0]
percReps	[4,7,10]
percRate	0.1
percSuccess	0.2
sojCleanse	[0.001,0.01,0.05,0.1,0.3,0.5,1.0]
sojReps	[4,7,10]
sojRate	0.1
sojSuccess	0.2
smCleanse	[0.001,0.01,0.05,0.1,0.3,0.5,1.0]
smReps	[4,7,10]
smRate	0.1
smSuccess	0.2
percEnable	[0, 1]
sojEnable	[0, 1]
smEnable	[0, 1]
falseHazardProb	1.0E-4
isRSSperfect	0
missHazardProb	0.05

Table 95: Intrusion Tolerance Sensitivity Analysis Parameters

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Cleansing=0.001	0.0445	0.9695	0.9992	1	1	1	1	1	1	1
Cleansing=0.01	0.0322	0.9424	0.9970	0.9999	1	1	1	1	1	1
Cleansing=0.05	0.0173	0.3765	0.6063	0.7508	0.8424	0.9003	0.9379	0.9605	0.9747	0.9837
Cleansing=0.1	0.0080	0.1183	0.2185	0.3042	0.3837	0.4533	0.5152	0.5703	0.6184	0.6619
Cleansing=0.3	0.0028	0.0320	0.0618	0.0897	0.1174	0.1438	0.1707	0.1954	0.2187	0.2416
Cleansing=0.5	0.0025	0.0274	0.0513	0.0744	0.0970	0.1189	0.1398	0.1600	0.1794	0.1989
Cleansing=1.0	0.0020	0.0234	0.0451	0.0657	0.0855	0.1060	0.1251	0.1437	0.1611	0.1796

Table 96: Intrusion Tolerance Sensitivity Analysis - Replica Cleansing (Raw Data)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Cleansing=0.001	0.0018	0.0015	2.5394e-04	7.7273e-10	7.7273e-10	7.7273e-10	7.7273e-10	7.7273e-10	7.7273e-10	7.7273e-10
Cleansing=0.01	0.0015	0.0020	4.8256e-04	1.0371e-04	7.7273e-10	7.7273e-10	7.7273e-10	7.7273e-10	7.7273e-10	7.7273e-10
Cleansing=0.05	0.0011	0.0042	0.0043	0.0038	0.0032	0.0026	0.0021	0.0017	0.0014	0.0011
Cleansing=0.1	7.7893e-04	0.0028	0.0036	0.0040	0.0043	0.0044	0.0044	0.0043	0.0043	0.0041
Cleansing=0.3	4.6482e-04	0.0015	0.0021	0.0025	0.0028	0.0031	0.0033	0.0035	0.0036	0.0038
Cleansing=0.5	4.4120e-04	0.0014	0.0019	0.0023	0.0026	0.0028	0.0030	0.0032	0.0034	0.0035
Cleansing=1.0	3.8768e-04	0.0013	0.0018	0.0022	0.0025	0.0027	0.0029	0.0031	0.0032	0.0034

Table 97: Intrusion Tolerance Sensitivity Analysis - Replica Cleansing (Confidence Levels)

As expected, the results (Table 96, Table 97) show that when changing the cleansing for the same number of replicas, increasing the cleansing removes the adversary faster giving less opportunity for the adversary to overcome enough replicas to compromise safety and security. Looking at these, I decided to choose a cleansing rate of 0.3 (3 times every 10 hours) as this would cleanse replicas at a good rate while still allowing visible observation of what areas have more of an effect of safety when compromised in the presence of such a defence.

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Replicas=10	0.0022	0.0227	0.0437	0.0634	0.0840	0.1039	0.1233	0.1423	0.1608	0.1800
Replicas=4	0.0028	0.0320	0.0618	0.0897	0.1174	0.1438	0.1707	0.1954	0.2187	0.2416
Replicas=7	0.0026	0.0260	0.0488	0.0717	0.0934	0.1149	0.1355	0.1553	0.1756	0.1958

Table 98: Intrusion Tolerance Sensitivity Analysis - Replicas (Raw Data)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Replicas=10	4.1068e-04	0.0013	0.0018	0.0021	0.0024	0.0027	0.0029	0.0031	0.0032	0.0034
Replicas=4	4.6482e-04	0.0015	0.0021	0.0025	0.0028	0.0031	0.0033	0.0035	0.0036	0.0038
Replicas=7	4.4978e-04	0.0014	0.0019	0.0023	0.0026	0.0028	0.0030	0.0032	0.0033	0.0035

Table 99: Intrusion Tolerance Sensitivity Analysis - Replicas (Confidence Levels)

For the results (Table 98, Table 99) when varying the number of replicas, a higher number of replicas gave lower risk of accident as there were more replicas to overcome. To look at the basic levels of safety and security that intrusion tolerance could offer, I chose 4 replicas as the value to use in the fixed model, as this seems plausible and shows the effectiveness of a basic intrusion tolerance policy.

7 Model Validation

This section is used to validate the models through a best effort approach with the help of my supervisors, by comparing relevant results from the models and their distributions where needed to make sure that the models are consistent with one another as new ones are made. The validation of these models uses the same simulation that the models themselves used (i.e. if a model was using numeric solver, this is what was used here too). It is extremely important to note that this validation relies on the assumption that the models from (Popov *et al.*, 2022) and (Kosari, Popov and Roy, 2022) are valid as the newer models are developed based off these so any validation attempts will always involve some sort of comparison with them.

7.1 Initial Models

To validate the multi-mode model, the single modes were compared with those of the initial model (albeit improved) version of the DESSERT'2022 to make sure the results are consistent with one another. The DESSERT'2022 version of the vulnerability mode in the tables below is named as DESSERT in each case while the equivalent vulnerability mode in the newer model is given its own name (i.e. the actual vulnerability mode name).

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
DESSERT	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734

Table 100: DESSERT vs. Multi-Mode Validation (Baseline)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
DESSERT	0.0022	0.0245	0.0462	0.0674	0.0882	0.1085	0.1283	0.1477	0.1677	0.1852
AccidentScale	0.0022	0.0245	0.0462	0.0674	0.0882	0.1085	0.1283	0.1477	0.1677	0.1852

Table 101: DESSERT vs. Multi-Mode Validation (AccidentScale)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
DESSERT	0.0045	0.0486	0.0907	0.1310	0.1695	0.2063	0.2415	0.2751	0.3072	0.3379
DelayScale	0.0045	0.0486	0.0907	0.1310	0.1695	0.2063	0.2414	0.2750	0.3071	0.3378

Table 102: DESSERT vs. Multi-Mode Validation (DelayScale)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
DESSERT	0.0021	0.0230	0.0434	0.0634	0.0829	0.1021	0.1209	0.1392	0.1572	0.1748
HazardScale	0.0021	0.0230	0.0434	0.0634	0.0829	0.1021	0.1209	0.1392	0.1572	0.1748

Table 103: DESSERT vs. Multi-Mode Validation (HazardScale)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
DESSERT	0.0021	0.0228	0.0431	0.0630	0.0824	0.1015	0.1201	0.1384	0.1563	0.1738
LateCH2AccScale	0.0021	0.0228	0.0431	0.0630	0.0824	0.1015	0.1201	0.1384	0.1563	0.1738

Table 104: DESSERT vs. Multi-Mode Validation (LateCH2AccScale)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
DESSERT	0.0021	0.0227	0.0429	0.0627	0.0821	0.1011	0.1197	0.1379	0.1558	0.1732
OH2CH scale	0.0021	0.0227	0.0429	0.0627	0.0821	0.1011	0.1197	0.1379	0.1558	0.1732

Table 105: DESSERT vs. Multi-Mode Validation (OH2CH_scale)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
DESSERT	0.0056	0.0600	0.1115	0.1601	0.2060	0.2495	0.2906	0.3294	0.3661	0.4008
AllModes	0.0056	0.0600	0.1114	0.1601	0.2060	0.2495	0.2905	0.3294	0.3661	0.4008

Table 106: DESSERT vs. Multi-Mode Validation (All Vulnerability Modes)

It is clear from Table 100-Table 106 that there is little to no difference between the comparable results of these two models, meaning that the new model with combinations of vulnerability modes is also valid and can be used in the next stage of the research.

7.2 Partitioned Model

In this section, the architecture-based model was validated by comparing the results of different partitions under attack with their vulnerability mode combinations in the prior model, which ensured that the new model was still consistent with the old one and had therefore correctly been transformed.

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline (Partitioned)	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
Baseline (Modes)	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734

Table 107: Architecture vs. Modes (Baseline)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Hazard (Partition)	0.0021	0.0230	0.0434	0.0634	0.0829	0.1021	0.1209	0.1392	0.1572	0.1748
HazardScale (Mode)	0.0021	0.0230	0.0434	0.0634	0.0829	0.1021	0.1209	0.1392	0.1572	0.1748

Table 108: Architecture vs. Modes (Hazard Perception)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
HazardOLH (Partition)	0.0054	0.0583	0.1084	0.1558	0.2007	0.2432	0.2835	0.3216	0.3576	0.3918
DelHazOH (Mode)	0.0054	0.0583	0.1084	0.1558	0.2007	0.2432	0.2835	0.3216	0.3576	0.3918

Table 109: Architecture vs. Modes (Hazard + Overlooked)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
HazardSM (Partitions)	0.0023	0.0245	0.0462	0.0675	0.0883	0.1086	0.1284	0.1479	0.1668	0.1854
AccHazLate (Modes)	0.0023	0.0245	0.0462	0.0675	0.0883	0.1086	0.1284	0.1479	0.1668	0.1854

Table 110: Architecture vs. Modes (Hazard + Safety Monitor)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Overlooked (Partition)	0.0024	0.0264	0.0497	0.0725	0.0947	0.1165	0.1377	0.1583	0.1785	0.1982
DelOH (Modes)	0.0024	0.0264	0.0497	0.0725	0.0947	0.1165	0.1377	0.1583	0.1785	0.1982

Table 111: Architecture vs. Modes (Overlooked Hazards)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
OverlookedSM (Partitions)	0.0026	0.0281	0.0530	0.0773	0.1009	0.1239	0.1464	0.1682	0.1895	0.2103
AccDelLateOH (Modes)	0.0026	0.0281	0.0530	0.0773	0.1009	0.1239	0.1464	0.1682	0.1895	0.2103

Table 112: Architecture vs. Modes (Overlooked + Safety Monitor)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
SafetyMonitor (Partition)	0.0023	0.0245	0.0463	0.0676	0.0884	0.1087	0.1286	0.1480	0.1670	0.1856
AccLate (Modes)	0.0023	0.0245	0.0463	0.0676	0.0884	0.1087	0.1286	0.1480	0.1670	0.1856

Table 113: Architecture vs. Modes (Safety Monitor)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
All (Partitions)	0.0056	0.0600	0.1114	0.1601	0.2060	0.2495	0.2905	0.3294	0.3661	0.4008
All (Modes)	0.0056	0.0600	0.1114	0.1601	0.2060	0.2495	0.2905	0.3294	0.3661	0.4008

Table 114: Architecture vs. Modes (All)

By comparing the results of the partitions against those of the modes (Table 107-Table 114), they are all identical meaning that the mode-based model has successfully been converted to the new architecture making it a valid model to be using from this stage onwards.

7.3 Uncertainty in Models

7.3.1 Lomax vs. Exponential (Time Intervals)

To ensure that the distribution applied to check for uncertainty is correct, it is important to validate the distributions to make sure they reach a similar expected value. This has been done by getting the firing times of timed activities used in the SAN models for both Lomax and Exponential distributions to get the time intervals to take the average of. From this, the expected values and confidence intervals have been obtained as shown in the table below. The full model documentations for these can be found in Appendix 13.6, 13.7 and 13.8.

Activity	Mean (Lomax)	Confidence (Lomax)	Mean (Exponential)	Confidence (Exponential)	Mean (Expected)
CompromisedReset	0.479299	0.037798937	4.94E-01	0.010712639	0.5
CorHaz2Acci (1)	0.00111946	8.82846E-05	1.15E-03	2.49893E-05	0.0012
CorHaz2Acci (2)	0.00111946	8.82843E-05	1.15E-03	2.49002E-05	0.0012
FalHaz2Acc	0.00111946	6.43901E-05	1.15E-03	2.58102E-05	0.0012
HDLateAcc (1)	0.00111946	6.43895E-05	1.15E-03	2.58094E-05	0.0012
HDLateAcc (2)	0.00111946	6.43895E-05	1.15E-03	2.58094E-05	0.0012
OK2FH	48.5612	2.793203898	5.01E+01	1.119932325	50.6586
OK2Hazard	0.0048566	0.000279347	5.01E-03	0.000112008	0.0051
Ready2Attack	9.58598	0.551342609	9.84E+00	0.219913769	1.00E+01
sojournTime (1)	0.000294953	1.6965E-05	3.04E-04	6.80317E-06	3.08E-04
sojournTime (2)	4.08E-05	2.34606E-06	4.21E-05	9.40666E-07	4.25E-05

Table 115: Lomax vs. Exponential Validation

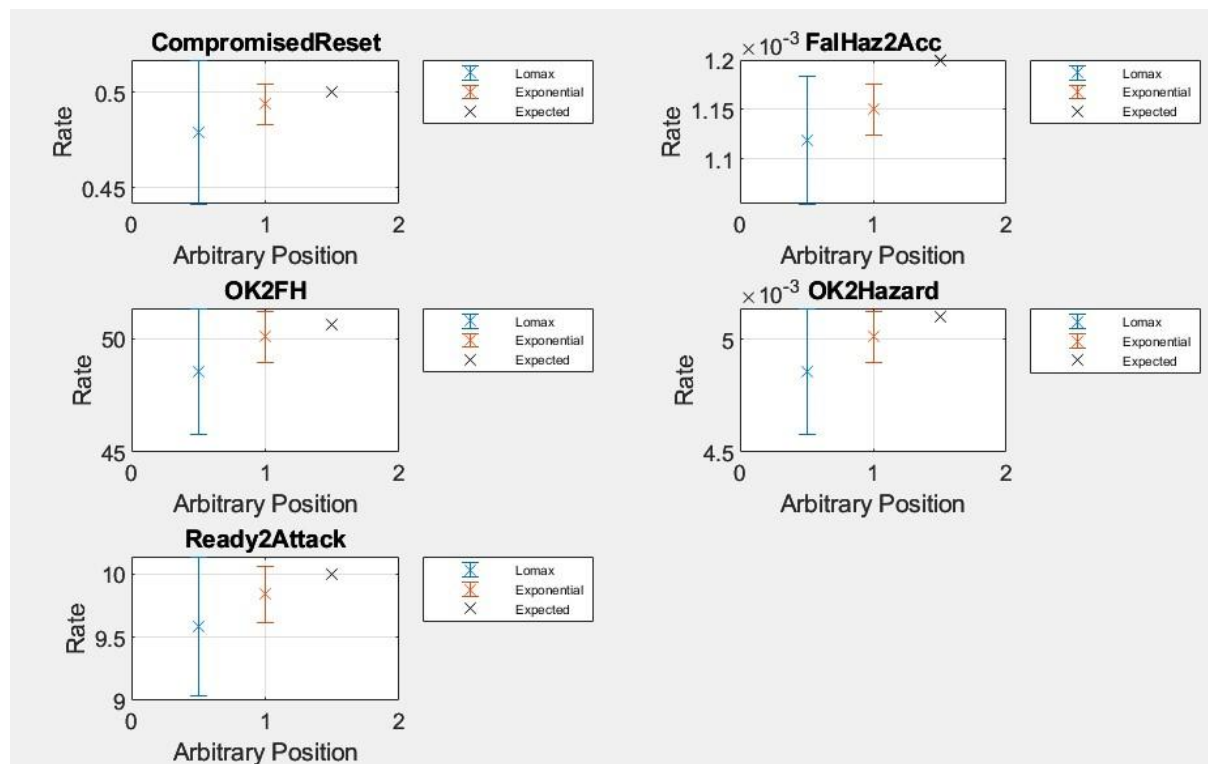


Figure 40: Lomax vs. Exponential Subplots (First Group)

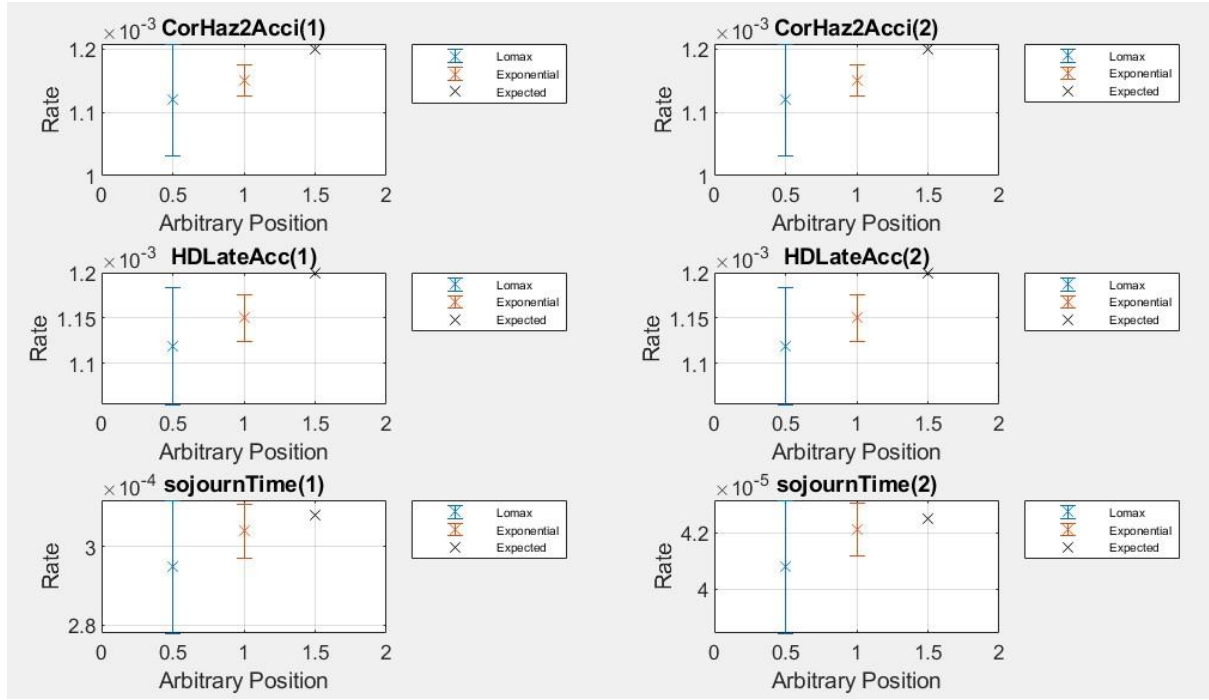


Figure 41: Lomax vs. Exponential Subplots (Second Group)

The information from Table 115 and its error bar plots Figure 40 and Figure 41 made for these validations, as shown in indicate that the expected value calculations for both distributions do overlap, which suggests they are in an acceptable range to one another even if they do not meet the expected value, which indicates that the expected values are essentially equal meaning that the distributions can be considered valid in this context.

7.4 Defence-in-Depth

This section validates the defence-in-depth by looking at the baselines compared to the regular partitioned model as well as the conditional probabilities to show that the product of the probabilities will still be the same regardless of their layer.

7.4.1 Baseline Comparison

As the model is obviously different due to the added layer which will give different results, there is little point in validating with the other combinations as they will be different. Therefore, the baseline was checked first to make sure that it works in the same way in the absence of attacks.

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Partition	0.0021	0.0228	0.0430	0.0628	0.0822a	0.1012	0.1198	0.1380	0.1559	0.1734
Defence-in-Depth	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734

Table 116: Defence-in-Depth Validation (Baseline Comparison)

Looking at Table 116, the baseline values of both models are shown to be the same which means that the model itself is correct even with the new defences such that the defences have not affected the natural state of the vehicle in the absence of an adversary.

7.4.2 Independence of Layers (Attacks)

This section aims to make use of the success probabilities of the two layers to show that they are independent of one another. This is done by changing one or more of the probabilities between the different layers, meaning the probability of successfully compromising the

system could be swapped with the probability of successfully compromising a particular subsystem to see if there is any change in the risk of accident, which should not be the case.

Parameter	Hazard Perception	Overlooked	Safety Monitor
AccidentScale	10.0	10.0	10.0
CH2AccRate	1.0E-4	1.0E-4	1.0E-4
CH2OKRate	856.3	856.3	856.3
DelayScale	1000.0	1000.0	1000.0
FH2AccRate	1.0E-5	1.0E-5	1.0E-5
FH2OKRate	856.3	856.3	856.3
HazardRate	197.4	197.4	197.4
HazardScale	10.0	10.0	10.0
LateCH2AccScale	10.0	10.0	10.0
MH2AccRate	0.001	0.001	0.001
MH2CHRate	2250.0	2250.0	2250.0
OH2OKRate	1000.0	1000.0	1000.0
OH2CH scale	10.0	10.0	10.0
attackRate	0.1	0.1	0.1
attackReset	2.0	2.0	2.0
attackSuccess	0.3	0.1	0.5
hazCleanseRate	0.001	0.001	0.001
hazSuccessProb	0.2	0.3	0.3
olhCleanseRate	0.001	0.001	0.001
olhSuccessProb	0.1	0.2	0.1
safeCleanseRate	0.001	0.001	0.001
safeSuccessProb	0.5	0.5	0.2
enableHazard	1	0	0
enableOLH	0	1	0
enableSM	0	0	1
falseHazardProb	1.0E-4	1.0E-4	1.0E-4
isRSSperfect	0	0	0
missHazardProb	0.05	0.05	0.05

Table 117: Defence-in-Depth - Conditional Probability - Success Probability Swap

In Table 117, this is how the probability is checked to see if it is conditional, by swapping the success probability of accessing the system (i.e. CAN bus) with the probability of compromising a certain subsystem once inside.

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
ConditionalHazard	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
Hazard	0.0024	0.0262	0.0474	0.0675	0.0870	0.1059	0.1245	0.1426	0.1603	0.1777

Table 118: Conditional Probability - Hazard Perception

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
ConditionalOLH	0.0053	0.0721	0.1041	0.1259	0.1449	0.1628	0.1802	0.1972	0.2138	0.2301
Overlooked	0.0070	0.0733	0.1044	0.1260	0.1449	0.1628	0.1802	0.1972	0.2138	0.2301

Table 119: Conditional Probability - Overlooked Hazards

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Baseline	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
ConditionalSM	0.0055	0.0500	0.0769	0.0983	0.1176	0.1361	0.1540	0.1716	0.1887	0.2055
SafetyMonitor	0.0051	0.0555	0.0842	0.1060	0.1254	0.1437	0.1616	0.1789	0.1959	0.2126

Table 120: Conditional Probability - Safety Monitor

Looking at the results (Table 118-Table 120), switching the probabilities has very small effects compared to one another, especially when swapping for that partition, which can be attributed to the cleansing rates on both layers still being the same throughout. This is because the product of the probabilities is still the same when swapped meaning the two events modelled by the respective probabilities are indeed independent.

7.5 Intrusion Tolerance

This section checks the baseline values of the intrusion tolerance model against all the previous models which used the same distributions to show that they are sufficiently close for it to be valid. This compares the modes, partitions, defence-in-depth as well as both static and dynamic versions of intrusion tolerance, which means this shows that all these models were equally valid on creation, as seen in the below tables, where the error bars due to the confidence also give sufficient overlap.

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Defence-in-Depth	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
Intrusion Tolerance (Dynamic)	0.0022	0.0225	0.0411	0.0617	0.0812	0.0995	0.1181	0.1365	0.1557	0.1727
Intrusion Tolerance (Static)	0.0022	0.0225	0.0411	0.0617	0.0812	0.0995	0.1181	0.1365	0.1557	0.1727
Modes	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734
Partitioned	0.0021	0.0228	0.0430	0.0628	0.0822	0.1012	0.1198	0.1380	0.1559	0.1734

Table 121: Intrusion Tolerance Validation - Baselines (Raw Data)

Experiment	100	1100	2100	3100	4100	5100	6100	7100	8100	9100
Intrusion Tolerance (Dynamic)	4.0694e-04	0.0013	0.0017	0.0021	0.0024	0.0026	0.0028	0.0030	0.0032	0.0033
Intrusion Tolerance (Static)	4.0694e-04	0.0013	0.0017	0.0021	0.0024	0.0026	0.0028	0.0030	0.0032	0.0033

Table 122: Intrusion Tolerance Validation - Baselines (Confidence Levels)

8 Discussions and Lessons Learned

This section discusses what I have found so far, and the lessons learned from the work. This will be discussed for each model and may help in forming the basis for future modelling for CAVs.

For the initial model (DESSERT'2022), it was found that the delaying of hazard perception was extremely dangerous on its own, rivalled only by deterioration of the safety monitor, suggesting these are two key areas to defend. However, it was also found that these were not the worst case, as compromising all of them was far worse, meaning that the next stage was to model combinations of different parts of the CAV being attacked.

When testing different combinations of modes under attack, attacking the perception (correct and timely detection of hazards) was found to have serious implications if done while the safety monitor integrity was also being affected. The overlooking of hazards was less of a concern as this tended to happen when the hazard was already not detected on time.

In the sensitivity analysis used to choose suitable values, the effect of false hazards (i.e. false alarms) was found to be negligible and therefore not a concern, meaning that this can either be ignored or set to a very low value just for the sake of not completely discounting it.

The partitioning of the model allowed for an architecture of sorts to be developed, in which compromising both elements of the perception (hazard and late/delay) would result in the greatest risk aside from compromising the entire vehicle system. Individually, late hazard recovery was the most severe, making it important to apply defences here.

For defence-in-depth, the delaying of hazard perception was once again a major issue, but the difference here was that the safety monitor was much more at risk due to higher successes in compromising it, suggesting that greater security measures must be taken to ensure the safety monitor works properly. This also showed that due to the hazard perception not having as much of a threat to safety meant that defence-in-depth worked well in protecting some elements of perception. The testing of both weak and strong adversaries in this model showed that defence-in-depth works well against both types.

When testing what were considered 'realistic' parameters (i.e. those obtained from attacks on computers) it was found that these had much lower risks of accident overall due to the significantly lower success probabilities of the adversary despite the higher rate of attack. However, it should be noted that these were not for attacks on a cyber-physical system and therefore may not be suitable for CAVs, so it would be better to obtain values from attacks on similar systems to CAVs where safety is a very real concern.

The uncertainty in the model parameters was checked by changing the distribution of the attack model between Exponential and Lomax, where the Lomax would account for any uncertainty. This was found to have very negligible results as both distributions gave very similar results. It should be noted that due to the Deterministic nature of the Lomax model and how sampling was done to obtain the rate values, the Exponential had to be modified to also sample from its own distribution in a similar way to avoid unnecessary inconsistencies.

From this, it was then found that the attack model can be improved in future work as the on/off for testing combinations that was being done to the actual vulnerability modes and partitions was better done at the attack model (i.e. enabling or disabling the attack itself rather

than its effects) which could allow for better modelling in future and avoid any possible interactions of the attack model with the CAV system when it is not desired.

For intrusion tolerance, it was found that choosing a good number of replicas can be an effective defence, provided a high cleansing rate is chosen. Dynamic or adaptive defences depending on the number of replicas were found not to be massively effective, especially if the adversary had the ability to vary attack intensity periodically, which would result in increased risk which was only partially mitigated if the cleansing rate were also dynamic. The order of risk of most combinations was the same as in the original partitioned model without defences where the replicas were dynamic, but when the model was fixed, compromising the safety monitor and late perception together was found to be a greater risk than just compromising perception components in combination. Overall, intrusion tolerance was found to be an effective defence and while still giving risk above the baseline, it was significantly reduced compared to having no defence present.

It must also be considered that this model is for a single vehicle, and all attacks are assumed to be coming from different places, so it does not fully represent the connected component of CAV (i.e. no more than a single vehicle in the model), which should be something implemented in future models to have at least two vehicles interacting so both their safety risks can be assessed when they interact with one another with or without attacks.

It must also be noted that while the model is limited to Exponentially distributed timed activities only, this is not a major issue as according to the findings of (Littlewood, 1979), all Markov models of this type are asymptotically Exponential no matter what distribution I choose to use for my timed activities, meaning that my results for the risk would give an Exponential result over time anyway.

The following limitations were also discovered during this research regarding the models:

- The absorbing state for accidents is always considered catastrophic which assumes that all safety implications are dangerous accidents when there could be multiple types of accident with different effects.
- Model assumes adversaries that occur at a fixed rate with some success probability, which is typically not 100% meaning that the model does not account for an adversary that can perfectly execute an attack every time without issue (i.e. I assume a limited level of expertise from the adversary).
- Only a limited number of defences were implemented, with only one of them applied across more than one layer, so this does not model the presence of multiple adversaries and countermeasures.
- Intrusion tolerance models use identical replicas as diversification of replicas was not modelled, which does not fully represent the full potential of intrusion tolerance as diversification would alter the attack surface for each replica, even changing the probability of compromising different replicas.
- The dynamic intrusion tolerance models do not have a variety of algorithms for implementing the proactive measures, which may not be the case and better algorithms may be used.
- The actual model is for a single autonomous vehicle, rather than for multiple vehicles interacting, which greatly restricts the extent to which the connected component of

CAVs is modelled, with the assumption being that my modelled attacks include those that involve attacks using other autonomous vehicles (i.e. through their communication channels or information transmitted).

9 Conclusions

The thesis addresses quantitative methods for combined safety and security analysis applied to CAV. The studies relied on a safety model developed by others (Popov *et al.*, 2022) for safety analysis of AV and expanded to include targeted attacks on different components of CAV.

The work itself has offered stochastic models of varying levels of abstraction for CAVs, with different attacks and defences implemented, allowing for assessment of cyber-resilience under different conditions and how this is linked to the safety of such vehicles.

Our findings suggest:

- Quantitative models allow one to rank different types of attacks according to their effect on AV safety and thus identify the parts of AV architecture which should be defended first.
- Quantitative models also allow one to study the effectiveness of defence strategies including defence in depth and intrusion tolerance with proactive recovery.
- I demonstrated that the analysis relies on parameters estimable for naturalistic driving or from lit sources and may require systematic sensitivity analysis for those parameters which are unknown in detail.
- I also studied the effect of parameter uncertainty (e.g. those related to attack rates) and demonstrated that the nature of the rate distributions has very limited effect on AV system safety.
- I demonstrated that proactive intrusion – tolerance (PIT) may be vulnerable to attacks of a competent adversary, who might be able to overcome the defence provided by a static PIT.
- I developed and demonstrated the effectiveness of a dynamic PIT, which adjust its configuration (degree of replication and intervals of cleansing) to the threats posed by the adversaries. The plausibility of the solution is also discussed in detail.
- False hazards are of little concern in CAVs, with the model not showing much difference when they were included or not.
- The perception of hazards being delayed is a very serious concern in autonomous vehicles which causes large increases in the risk of accident, so this is where defences should be applied first. Experimental data indicated that perception was the most vulnerable area in CAVs.
- The safety monitor is very vulnerable even in the presence of defence-in-depth and should therefore have stronger defences (i.e. higher cleansing or a different type of defence)
- The uncertainty in model parameters can be considered negligible, so the use of the Exponential distribution in modelling is perfectly fine. Their relationship as part of a semi-Markov model can be considered asymptotic.

- Intrusion tolerance is a quite effective defence, even slightly better than defence-in-depth, but the dynamic variations of it are found to make it worse, especially if the adversary can vary their intensity. Cleansing rate is a very key element, suggesting that the response rate to an attack is very important in CAVs.

Looking at the work from an industrial standpoint, I determined that the implications for the transport section are as follows:

1. Layered and robust defences are required to adequately preserve safety and security in autonomous vehicles.
2. Due to the unpredictable nature of adversaries, defence needs to include some level of monitoring as well as dynamic or adaptive approaches that can account for a range of adversaries.
3. The linking of vehicles and cyberspace in CAVs means that experts in both the cyber and transport sectors must collaborate on vehicle projects to ensure safe and secure vehicle design.

To reiterate, the gaps addressed and the contributions, original or otherwise, were as follows:

1. **How will CAVs be modelled to quantify safety in the presence of cyber-attacks?**
Contributions: The work offers specific and detailed models of how successful attacks on CAVs affect failure modes and accounts for uncertainty in model parameters. The models reflect a CAV architecture including elements such as the perception and safety monitor, building on similar cyber-physical systems developed in prior works by my supervisors.
2. **How useful are these quantitative models, given that they rely on numerous parameters and what are their limitations?**
Contributions: I address the issue of obtaining plausible model parameters that will reflect CAV in a realistic manner and validate the models against previous models. The impact of uncertainty in different parameters is also studied.
3. **How is the proposed methodology to assess CAV safety in the presence of cyber-attacks in line with realistic CAV architectures?**
Contributions: The level of abstraction of the model is changed as I progress, creating one that groups the failure modes such that I have ‘partitions’ to reflect different components such as the safety monitor, giving an architecture in which components are linked and can be attacked separately or together.
4. **What methodology can be used to assess the safety impact of simultaneous attacks on multiple CAV components?**
Contributions: The model is designed such that components can be attacked individually and separately (i.e. enabling and disabling attack effects on components) to compare the effects, which is significant in originality as similar studies have not been conducted before, with an early version of it having been published in DESSERT’2022. Later development also allowed for non-linear behaviour to be observed when subjecting multiple components to attack.
5. **Considering the research questions so far, can the newer methodologies be applied to layered models of defence such as defence-in-depth?**

Contributions: The methodologies used are applied specifically to the CAV domain and are applicable to a wider range of cyber-resilient, cyber-physical systems beyond just CAV (e.g. UAVs).

6. Can a methodology be developed for quantitative modelling of intrusion tolerant CAVs?

Contributions: The development of CAV models where intrusion tolerance is applied to each of the components. This is then extended to have a more dynamic approach (i.e. proactive intrusion tolerance) as both literature and experimentation suggested that a static approach cannot account for future adversaries that may be unpredictable. The mechanisms for defeating dynamic intrusion tolerance have also been observed with a view to further publication on this work.

10 Further Work

Methodologically the work is based on the core idea that compromised software becomes less reliable (Popov, 2017). Although this view is quite plausible and consistent with the empirical studies confirming that compromised software does fail more frequently (including some extreme cases such as “Stuxnet”), the way of modelling reliability deterioration via a “scaling factor” may be questionable. An alternative and widely spread view is to consider the compromised software as not fit for purpose, hence any analysis beyond the point of a successful compromise would be of limited values. I disagree with such an extreme viewpoint but accept that the current study assumes that the compromised software will continue its operation post compromise. For those cases when the “game is over” immediately after a successful attack, this work would be of limited value.

Several variants of a model of CAV are used with different degree of details – from an abstract model of an AV to an AV with proactive intrusion detections applied either to the entire system (the entire stack of an AV appears as a monolith replica to replication applied to the level of individual ECU components). All models rely on a set of parameters which must be estimated. While the modelling approach is generic and applicable to any level of detail, the results, obtained after specific parameterisation, are limited to those set of parameter values and do not cover cases where other set of parameters are needed.

I also noted that the level of detail of CAV operational environment these days is seen as subject to operational design domain (ODD) (BSI, 2020), which is not explicitly addressed in the current work. Accounting for ODD would require extending the states which capture hazards to match the number of different operating conditions included in an ODD. This extension is straight forward but will require significant increase of model complexity and the number of probabilistic parameters to capture the CAV hazards and risks of accidents in them. Such an extension will be conducted in the future.

The modelling of the CAV is done for a single vehicle which could be considered a limitation as any attacks from other vehicles are more of an assumption than a fact of the model, meaning that future models should be developed that will have more than one vehicle interacting to assess the safety implications of poor security in both the ego vehicle and its surrounding vehicles.

The parameterization while plausible for the vehicle is not a true representation of the current vehicles on the road and despite attempts to find realistic values, these did not represent cyber-physical systems meaning that future models will need access to datasets from actual vehicles to properly model and validate the work with plausible values for the risk to safety of CAVs.

For intrusion tolerance, more detailed modelling and parameterization are definitely a good idea, with the possibility of introducing dynamic models that focus on cleansing rather than replicas or adversaries. It may be a good idea to develop systems that use dynamic cleansing (i.e. improved response rates) as a reaction to an adversary and to test more models where an adversary has more dynamic and unpredictable behaviour in different circumstances. More detailed replica diversification as well as other algorithms for the dynamic models could also be introduced as part of future work.

For the CAV, the level of abstraction is quite high and should be remodelled to reflect present architectures including all the ECUs and the computer systems that are in use, as well as the future models which make use of automotive ethernet to assess the safety of those vehicle architectures as well, which would mean a more detailed model with extra parameters.

It may be possible to implement the defences both system-wide and at the component level, trying to look at how effective they are compared to one another as well as comparing the level of overhead they may have when being deployed.

The model is sufficiently flexible and abstract to the point where it can be extended to other autonomous systems such as drones, which would only require some adjustment of parameters to reflect the new dimensions introduced by flight.

The models themselves could also be refined to include multiple vehicles and the impact of compromising one vehicle on other vehicles, both in proximity and far away. It may also be possible to consider other probabilistic distributions (i.e. to change behaviour of different elements where needed) and expanding on the 'Accident' state to reflect the different types of accidents that could occur.

11 References

- Ahn, H., Choi, J. and Kim, Y.H. (2023) ‘A Mathematical Modeling of Stuxnet-Style Autonomous Vehicle Malware’, *IEEE Transactions on Intelligent Transportation Systems*, 24(1), pp. 673–683. Available at: <https://doi.org/10.1109/TITS.2022.3213771>.
- Alam, M.S.U. *et al.* (2019) ‘Securing Vehicle ECU Communications and Stored Data’, in. Available at: <https://doi.org/10.1109/ICC.2019.8762043>.
- Alawieh, M. *et al.* (2022) ‘A Secure Scheme for Vehicle-to-Vehicle (V2V) Routing Protocol’, in *2022 5th Conference on Cloud and Internet of Things (CIoT). 2022 5th Conference on Cloud and Internet of Things (CIoT)*, pp. 1–8. Available at: <https://doi.org/10.1109/CIoT53061.2022.9766544>.
- Aliwa, E. *et al.* (2020) ‘Cyberattacks and Countermeasures For In-Vehicle Networks’. arXiv. Available at: <http://arxiv.org/abs/2004.10781> (Accessed: 22 August 2022).
- Bessani, A.N. (2011) ‘From Byzantine fault tolerance to intrusion tolerance (a position paper)’, in *2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W). 2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pp. 15–18. Available at: <https://doi.org/10.1109/DSNW.2011.5958857>.
- Bloomfield, R. *et al.* (2021) ‘Safety Case Templates for Autonomous Systems’. arXiv. Available at: <https://doi.org/10.48550/arXiv.2102.02625>.
- BSI (2018a) ‘Cyber risk and resilience - Guidance for the governing body and executive management’. BSI Group.
- BSI (2018b) ‘Road vehicles - Functional safety’. BSI Group.
- BSI (2018c) ‘The fundamental principles of automotive cyber security - Specification’. BSI Group.
- BSI (2020) ‘Operational Design Domain (ODD) Taxonomy for ADS Specification’. BSI Group. Available at: <https://www.bsigroup.com/en-GB/insights-and-media/insights/brochures/pas-1883-operational-design-domain-odd-taxonomy-for-ads-specification/>.
- BSI (2021) ‘Road vehicles - Cybersecurity engineering’. BSI Group.
- BSI (2022) ‘Road vehicles - Safety of the intended functionality’. BSI Group.
- Cao, Y. *et al.* (2022) ‘You Can’t See Me: Physical Removal Attacks on LiDAR-based Autonomous Vehicles Driving Frameworks’. arXiv. Available at: <http://arxiv.org/abs/2210.09482> (Accessed: 2 November 2022).
- Choi, C.Q. (2021) ‘7 Revealing Ways AIs Fail’, *IEEE Spectrum*, 58(10), pp. 42–47. Available at: <https://doi.org/10.1109/MSPEC.2021.9563958>.
- Chowdhury, M., Islam, M. and Khan, Z. (2019) ‘Security of Connected and Automated Vehicles’, 49, pp. 46–56.

- CPC (2019) *Connected Places Catapult, Connected Places Catapult*. Available at: <https://cp.catapult.org.uk/> (Accessed: 17 October 2022).
- Crawley, F. and Tyler, B. (2015) *HAZOP: guide to best practice: guidelines to best practice for the process and chemical industries*. Third edition. Amsterdam: Elsevier.
- Cui, J. *et al.* (2019) ‘A review on safety failures, security attacks, and available countermeasures for autonomous vehicles’, *Ad Hoc Networks*, 90(101823). Available at: <https://doi.org/10.1016/j.adhoc.2018.12.006>.
- Culley, J. *et al.* (2020) ‘System Design for a Driverless Autonomous Racing Vehicle’, in *2020 12th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP). 2020 12th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, Porto, Portugal: IEEE, pp. 1–6. Available at: <https://doi.org/10.1109/CSNDSP49049.2020.9249626>.
- Dibaei, M. *et al.* (2019) ‘An Overview of Attacks and Defences on Intelligent Connected Vehicles’. arXiv. Available at: <https://doi.org/10.48550/arXiv.1907.07455>.
- El-Rewini, Z. *et al.* (2020) ‘Cybersecurity challenges in vehicular communications’, *Vehicular Communications*, 23, p. 100214. Available at: <https://doi.org/10.1016/j.vehcom.2019.100214>.
- Favaro, F. *et al.* (2023) ‘Building a Credible Case for Safety: Waymo’s Approach for the Determination of Absence of Unreasonable Risk’. arXiv. Available at: <https://doi.org/10.48550/arXiv.2306.01917>.
- Ford, M.D. *et al.* (2013) ‘Implementing the ADVISE security modeling formalism in Möbius’, in *2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). 2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 1–8. Available at: <https://doi.org/10.1109/DSN.2013.6575362>.
- Forrest, G. (2010) ‘FMEA (Failure Mode and Effects Analysis) Quick Guide’, *isixsigma.com*, 31 December. Available at: <https://bigscoots.isixsigma.com/uncategorized/fmea-quick-guide/> (Accessed: 3 February 2023).
- Forsberg, A. and Hedberg, J. (2009) ‘Comparison of FlexRay and CAN-bus for Real-Time Communication’. Available at: https://www.academia.edu/44171258/Comparison_of_FlexRay_and_CAN_bus_for_Real_Time_Communication (Accessed: 24 January 2023).
- Foster, I. *et al.* (2015) ‘Fast and Vulnerable: A Story of Telematic Failures’, in *9th USENIX Workshop on Offensive Technologies (WOOT 15)*. Available at: <https://www.usenix.org/conference/woot15/workshop-program/presentation/foster> (Accessed: 19 August 2022).
- Fowler, D.S. (2020) *Project BeARCAT: Baselineing, Automation and Response for CAV Testbed Cyber Security: Connected Vehicle & Infrastructure Security Assessment*. Report. London: Zenzic. Available at: <http://wrap.warwick.ac.uk/141491/> (Accessed: 25 January 2023).

- Galdino, S. and Maciel, P. (2006) ‘Interval Generalized Stochastic Petri Net Models in Performance Evaluation’, in *2006 IEEE International Conference on Systems, Man and Cybernetics. 2006 IEEE International Conference on Systems, Man and Cybernetics*, pp. 2723–2728. Available at: <https://doi.org/10.1109/ICSMC.2006.385285>.
- Gao, C. *et al.* (2022) ‘Autonomous Driving Security: State of the Art and Challenges’, *IEEE Internet of Things Journal*, 9(10), pp. 7572–7595. Available at: <https://doi.org/10.1109/JIOT.2021.3130054>.
- Girdhar, M. *et al.* (2022) ‘Post-Accident Cyberattack Event Analysis for Connected and Automated Vehicles’, *IEEE Access*, 10, pp. 83176–83194. Available at: <https://doi.org/10.1109/ACCESS.2022.3196346>.
- Gozubuyuk, B. *et al.* (2023) ‘An Overview of Security in Connected and Autonomous Vehicles’, in *2023 International Conference on Artificial Intelligence of Things and Systems (AIoTSys). 2023 International Conference on Artificial Intelligence of Things and Systems (AIoTSys)*, pp. 206–213. Available at: <https://doi.org/10.1109/AIoTSys58602.2023.00052>.
- Greenberg, A. (2015) ‘Hackers Remotely Kill a Jeep on the Highway—With Me in It’, *Wired*, 21 July. Available at: <https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/> (Accessed: 20 February 2023).
- Grzemba, A. (ed.) (2011) *MOST: the automotive multimedia network ; from MOST25 to MOST150*. 2. ed. Poing: Franzis (Electronics library). Available at: http://www2.ciando.com/img/books/extract/3645250611_lp.pdf.
- Gupta, S., Maple, C. and Passerone, R. (2023) ‘An Investigation of Cyber-Attacks and Security Mechanisms for Connected and Autonomous Vehicles’, *IEEE Access*, 11, pp. 90641–90669. Available at: <https://doi.org/10.1109/ACCESS.2023.3307473>.
- Hagiescu, A. *et al.* (2007) ‘Performance Analysis of FlexRay-based ECU Networks’, in *2007 44th ACM/IEEE Design Automation Conference. 2007 44th ACM/IEEE Design Automation Conference*, pp. 284–289. Available at: <https://ieeexplore.ieee.org/document/4261192>.
- Hasbullah, H., Soomro, I.A. and Manan, J.A. (2010) ‘Denial of Service (DOS) Attack and Its Possible Solutions in VANET’, *International Journal of Electronics and Communication Engineering*, 4(5), pp. 813–817.
- van der Heijden, R.W., Lukaseder, T. and Kargl, F. (2018) ‘VeReMi: A Dataset for Comparable Evaluation of Misbehavior Detection in VANETs’. arXiv. Available at: <http://arxiv.org/abs/1804.06701> (Accessed: 20 June 2023).
- IEEE Standards Association (2022) *IEEE Standard for Assumptions in Safety-Related Models for Automated Driving Systems*. IEEE. Available at: <https://doi.org/10.1109/IEEESTD.2022.9761121>.
- IET (2021) ‘Code of Practice - Cybersecurity and Safety’. IET. Available at: https://electrical.theiet.org/media/2516/cop_cyber-security-and-safety_linkable_secure.pdf.
- Iqbal, S. *et al.* (2022) ‘Simulating Malicious Attacks on VANETs for Connected and Autonomous Vehicle Cybersecurity: A Machine Learning Dataset’, in *2022 13th International Symposium on Communication Systems, Networks and Digital Signal*

Processing (CSNDSP). 2022 13th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP), pp. 332–337. Available at: <https://doi.org/10.1109/CSNDSP54353.2022.9908023>.

Jafarnejad, S. *et al.* (2015) ‘A Car Hacking Experiment: When Connectivity Meets Vulnerability’, in *2015 IEEE Globecom Workshops (GC Wkshps)*. 2015 IEEE Globecom Workshops (GC Wkshps), pp. 1–6. Available at: <https://doi.org/10.1109/GLOCOMW.2015.7413993>.

Jonnalagadda, S.K. and Jakkala, P. (2020) ‘Cyber Security and Risk Analysis on Connected Autonomous Vehicles’, *Solid State Technology*, 63, pp. 10161–10176.

Kamel, J. *et al.* (2020) ‘VeReMi Extension: A Dataset for Comparable Evaluation of Misbehavior Detection in VANETs’, in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*. ICC 2020 - 2020 IEEE International Conference on Communications (ICC), Dublin, Ireland: IEEE, pp. 1–6. Available at: <https://doi.org/10.1109/ICC40277.2020.9149132>.

Kay, G. (2023) *Video shows 8-car pileup after a Tesla allegedly using Full Self-Driving stopped in a highway tunnel*, *Business Insider*. Available at: <https://www.businessinsider.com/tesla-stops-tunnel-pileup-accidents-driver-says-fsd-enabled-video-2023-1> (Accessed: 20 February 2023).

Khan, W.Z., Khan, M.K. and Arshad, Q.-A. (2023) ‘Cybersecurity Digital Labels for Connected and Autonomous Vehicles’, *IEEE Consumer Electronics Magazine*, 12(1), pp. 87–93. Available at: <https://doi.org/10.1109/MCE.2021.3119857>.

Kitchenham, B. and Brereton, P. (2013) ‘A systematic review of systematic review process research in software engineering’, *Information and Software Technology*, 55(12), pp. 2049–2075. Available at: <https://doi.org/10.1016/j.infsof.2013.07.010>.

Koopman, P. (2022) *How Safe is Safe Enough?: Measuring and Predicting Autonomous Vehicle Safety*. Carnegie Mellon University.

Kopetz, H. *et al.* (2005) ‘The time-triggered Ethernet (TTE) design’, in *Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC’05)*. Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC’05), pp. 22–33. Available at: <https://doi.org/10.1109/ISORC.2005.56>.

Kosari, A., Popov, P. and Roy, R. (2022) ‘Modelling Safety of Connected and Autonomous Vehicles (CAVs) under Cyber-Attacks on Perception and Safety Monitors’, in *2022 12th International Conference on Dependable Systems, Services and Technologies (DESSERT)*. 2022 12th International Conference on Dependable Systems, Services and Technologies (DESSERT), pp. 1–7. Available at: <https://doi.org/10.1109/DESSERT58054.2022.10018781>.

Koscher, K. *et al.* (2010) ‘Experimental Security Analysis of a Modern Automobile’, in *2010 IEEE Symposium on Security and Privacy*. 2010 IEEE Symposium on Security and Privacy, Oakland, CA, USA: IEEE, pp. 447–462. Available at: <https://doi.org/10.1109/SP.2010.34>.

Kumar, A. and Mehta, S. (2017) ‘A Survey on Resilient Machine Learning’. arXiv. Available at: <https://doi.org/10.48550/arXiv.1707.03184>.

Kwiatkowska, M., Norman, G. and Parker, D. (2011) ‘PRISM 4.0: Verification of Probabilistic Real-Time Systems’, in G. Gopalakrishnan and S. Qadeer (eds) *Computer Aided Verification*. Berlin, Heidelberg: Springer Berlin Heidelberg (Lecture Notes in Computer Science), pp. 585–591. Available at: https://doi.org/10.1007/978-3-642-22110-1_47.

Levermore, T. and Peters, A. (2020) ‘Test framework and key challenges for virtual verification of automated vehicles: the VeriCAV project’, in *39th International Conference on Computer Safety, reliability and Security (SAFECOMP), Position Paper*. Lisbon, Portugal. Available at: <https://hal.science/hal-02931723> (Accessed: 7 January 2024).

Li, A., Wang, J. and Zhang, N. (2021) ‘Chronos: Timing Interference as a New Attack Vector on Autonomous Cyber-physical Systems’, in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery (CCS ’21), pp. 2426–2428. Available at: <https://doi.org/10.1145/3460120.3485350>.

Lisova, E., Sljivo, I. and Causevic, A. (2019) ‘Safety and Security Co-Analyses: A Systematic Literature Review’, *IEEE Systems Journal*, 13(3), pp. 2189–2200. Available at: <https://doi.org/10.1109/JSYST.2018.2881017>.

Littlewood, B. (1979) ‘Software Reliability Model for Modular Program Structure’, *IEEE Transactions on Reliability*, R-28(3), pp. 241–246. Available at: <https://doi.org/10.1109/TR.1979.5220576>.

Liu, M. *et al.* (2019) ‘An Efficient Scheduling Algorithm for Adjustable Time Slots in Time-Triggered Ethernet’, in *2019 IEEE 11th International Conference on Communication Software and Networks (ICCSN). 2019 IEEE 11th International Conference on Communication Software and Networks (ICCSN)*, pp. 68–72. Available at: <https://doi.org/10.1109/ICCSN.2019.8905355>.

Loveless, A. *et al.* (2023) ‘PCSPOOF: Compromising the Safety of Time-Triggered Ethernet’, in *2023 IEEE Symposium on Security and Privacy (SP). 2023 IEEE Symposium on Security and Privacy (SP)*, pp. 3193–3208. Available at: <https://doi.org/10.1109/SP46215.2023.10179318>.

Lu, Z., Qu, G. and Liu, Z. (2019) ‘A Survey on Recent Advances in Vehicular Network Security, Trust, and Privacy’, *IEEE Transactions on Intelligent Transportation Systems*, 20(2), pp. 760–776. Available at: <https://doi.org/10.1109/TITS.2018.2818888>.

Malik, S. and Sun, W. (2020) ‘Analysis and Simulation of Cyber Attacks Against Connected and Autonomous Vehicles’, in *2020 International Conference on Connected and Autonomous Driving (MetroCAD). 2020 International Conference on Connected and Autonomous Driving (MetroCAD)*, pp. 62–70. Available at: <https://doi.org/10.1109/MetroCAD48866.2020.00018>.

Mariotti, F. *et al.* (2023) ‘An extension of the ADVISE Meta modeling framework and its application for an early-stage security analysis of a public transport supervision system’, *Journal of Reliable Intelligent Environments*, 9(3), pp. 263–281. Available at: <https://doi.org/10.1007/s40860-023-00209-5>.

- Mostaq Hossain, S.M. *et al.* (2023) ‘Survey on Security Attacks in Connected and Autonomous Vehicular Systems’, in *2023 IEEE International Conference on Computing (ICOCO)*. *2023 IEEE International Conference on Computing (ICOCO)*, pp. 295–300. Available at: <https://doi.org/10.1109/ICOCO59262.2023.10397929>.
- Myklebust, T., Stålhane, T. and Jenssen, G.D. (2023) ‘Autonomous Vehicles - Trust, Safety and Security Cases: The Complete Picture’, in *2023 Annual Reliability and Maintainability Symposium (RAMS)*. *2023 Annual Reliability and Maintainability Symposium (RAMS)*, pp. 1–6. Available at: <https://doi.org/10.1109/RAMS51473.2023.10088202>.
- Nayak, B.P. *et al.* (2022) ‘Autonomous Vehicles: Resource Allocation, Security, and Data Privacy’, *IEEE Transactions on Green Communications and Networking*, 6(1), pp. 117–131. Available at: <https://doi.org/10.1109/TGCN.2021.3110822>.
- Netkachov, O., Popov, P.T. and Salako, K. (2019) ‘Quantitative Evaluation of the Efficacy of Defence-in-Depth in Critical Infrastructures’, in. Berlin, Germany: Springer International Publishing, pp. 89–121. Available at: https://doi.org/10.1007/978-3-319-95597-1_5.
- Nikishin, K. and Konnov, N. (2020) ‘Schedule Time-Triggered Ethernet’, in *2020 International Conference on Engineering Management of Communication and Technology (EMCTECH)*. *2020 International Conference on Engineering Management of Communication and Technology (EMCTECH)*, pp. 1–5. Available at: <https://doi.org/10.1109/EMCTECH49634.2020.9261540>.
- Oboril, F. and Scholl, K.-U. (2021) ‘RSS+: Pro-Active Risk Mitigation for AV Safety Layers based on RSS’, in *2021 IEEE Intelligent Vehicles Symposium (IV)*. *2021 IEEE Intelligent Vehicles Symposium (IV)*, pp. 99–106. Available at: <https://doi.org/10.1109/IV48863.2021.9575731>.
- Pal, P. *et al.* (2006) ‘An architecture for adaptive intrusion-tolerant applications’, *Software: Practice and Experience*, 36(11–12), pp. 1331–1354. Available at: <https://doi.org/10.1002/spe.747>.
- Parkinson, S. *et al.* (2017) ‘Cyber Threats Facing Autonomous and Connected Vehicles: Future Challenges’, *IEEE Transactions on Intelligent Transportation Systems*, 18(11), pp. 2898–2915. Available at: <https://doi.org/10.1109/TITS.2017.2665968>.
- Peng, Z. *et al.* (2024) ‘Security Analysis for Autonomous Ground Vehicle Under Stealthy Sensor Attacks’, *IEEE Transactions on Vehicular Technology*, 73(9), pp. 12426–12437. Available at: <https://doi.org/10.1109/TVT.2024.3387737>.
- Pham, M. and Xiong, K. (2020) ‘A Survey on Security Attacks and Defense Techniques for Connected and Autonomous Vehicles’. arXiv. Available at: <https://doi.org/10.48550/arXiv.2007.08041>.
- Platania, M. *et al.* (2014) ‘Towards a Practical Survivable Intrusion Tolerant Replication System’, in *2014 IEEE 33rd International Symposium on Reliable Distributed Systems*. *2014 IEEE 33rd International Symposium on Reliable Distributed Systems*, pp. 242–252. Available at: <https://doi.org/10.1109/SRDS.2014.16>.
- Popov, P. (2017) ‘Models of Reliability of Fault-Tolerant Software Under Cyber-Attacks’, in *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*. *2017*

IEEE 28th International Symposium on Software Reliability Engineering (ISSRE), pp. 228–239. Available at: <https://doi.org/10.1109/ISSRE.2017.23>.

Popov, P.T. (2015) ‘Stochastic Modeling of Safety and Security of the e-Motor, an ASIL-D Device’, in F. Koornneef and C. van Gulijk (eds). *34th International Conference on Computer Safety, Reliability, and Security, SAFECOMP 2015*, Delft University of Technology, Netherlands. Available at: <https://openaccess.city.ac.uk/id/eprint/12518/> (Accessed: 14 November 2022).

Popov, P.T. *et al.* (2022) ‘Modelling road hazards and the effect on AV safety of hazardous failures’, in. *The 25th IEEE International Conference on Intelligent Transportation Systems (IEEE ITSC 2022)*, Macau, China. Available at: <https://openaccess.city.ac.uk/id/eprint/28344/> (Accessed: 14 November 2022).

Qayyum, A. *et al.* (2020) ‘Securing Connected & Autonomous Vehicles: Challenges Posed by Adversarial Machine Learning and the Way Forward’, *IEEE Communications Surveys & Tutorials*, 22(2), pp. 998–1026. Available at: <https://doi.org/10.1109/COMST.2020.2975048>.

Ramsbrock, D., Berthier, R. and Cukier, M. (2007) ‘Profiling Attacker Behavior Following SSH Compromises’, in *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN’07)*. *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN’07)*, pp. 119–124. Available at: <https://doi.org/10.1109/DSN.2007.76>.

Riffee, A.A., Riffee, A. and Youssef, T.A. (2024) ‘Connected Vehicle Security: Overview of Risks and Mitigation Strategies’, in *SoutheastCon 2024*. *SoutheastCon 2024*, pp. 1432–1437. Available at: <https://doi.org/10.1109/SoutheastCon52093.2024.10500211>.

Ross, R., McEvilly, M. and Carrier Oren, J. (2016) *Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems*. NIST SP 800-160. National Institute of Standards and Technology, p. NIST SP 800-160. Available at: <https://doi.org/10.6028/NIST.SP.800-160>.

SAE (2021) ‘Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles’. SAE.

Sanders, W.H. and Applequist, J. (2012a) *ADVISE Atomic Formalism - Mobius Wiki*. Available at: https://www.mobius.illinois.edu/wiki/index.php/ADVISE_Atomic_Formalism (Accessed: 15 November 2025).

Sanders, W.H. and Applequist, J. (2012b) *The Möbius Tool*. Available at: <http://www.mobius.illinois.edu/> (Accessed: 2 November 2022).

Sanders, W.H. and Meyer, J.F. (2001) ‘Stochastic Activity Networks: Formal Definitions and Concepts★’, in E. Brinksma, H. Hermanns, and J.-P. Katoen (eds) *Lectures on Formal Methods and Performance Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg (Lecture Notes in Computer Science), pp. 315–343. Available at: https://doi.org/10.1007/3-540-44667-2_9.

Scalas, M. and Giacinto, G. (2019) ‘Automotive Cybersecurity: Foundations for Next-Generation Vehicles’. arXiv. Available at: <https://doi.org/10.48550/arXiv.1910.01037>.

Schmittner, C. *et al.* (2015) ‘A case study of FMVEA and CHASSIS as safety and security co-analysis method for automotive cyber-physical systems’, *CPSS 2015 - Proceedings of the 1st ACM Workshop on Cyber-Physical System Security, Part of ASIACCS 2015*, pp. 69–80. Available at: <https://doi.org/10.1145/2732198.2732204>.

Schmittner, C., Ma, Z. and Smith, P. (2014) ‘FMVEA for Safety and Security Analysis of Intelligent and Cooperative Vehicles’, in A. Bondavalli, A. Ceccarelli, and F. Ortmeier (eds) *Computer Safety, Reliability, and Security*. Cham: Springer International Publishing (Lecture Notes in Computer Science), pp. 282–288. Available at: https://doi.org/10.1007/978-3-319-10557-4_31.

Sedar, R. *et al.* (2023) ‘A Comprehensive Survey of V2X Cybersecurity Mechanisms and Future Research Paths’, *IEEE Open Journal of the Communications Society*, 4, pp. 325–391. Available at: <https://doi.org/10.1109/OJCOMS.2023.3239115>.

Shalev-Shwartz, S., Shammah, S. and Shashua, A. (2018) ‘On a Formal Model of Safe and Scalable Self-driving Cars’. arXiv. Available at: <http://arxiv.org/abs/1708.06374> (Accessed: 14 November 2022).

Shangguan, L. *et al.* (2022) ‘Dynamic Watermarking for Cybersecurity of Autonomous Vehicles’, *IEEE Transactions on Industrial Electronics*, pp. 1–8. Available at: <https://doi.org/10.1109/TIE.2022.3229333>.

Sharma, P. and Gillanders, J. (2022) ‘Cybersecurity and Forensics in Connected Autonomous Vehicles: A Review of the State-of-the-Art’, *IEEE Access*, 10, pp. 108979–108996. Available at: <https://doi.org/10.1109/ACCESS.2022.3213843>.

Sharma, P. and Möller, D.P.F. (2018) ‘Protecting ECUs and Vehicles Internal Networks’, in *2018 IEEE International Conference on Electro/Information Technology (EIT)*. *2018 IEEE International Conference on Electro/Information Technology (EIT)*, pp. 0465–0470. Available at: <https://doi.org/10.1109/EIT.2018.8500295>.

Shirvani, S., Baseri, Y. and Ghorbani, A. (2024) ‘Evaluation Framework for Electric Vehicle Security Risk Assessment’, *IEEE Transactions on Intelligent Transportation Systems*, 25(1), pp. 33–56. Available at: <https://doi.org/10.1109/TITS.2023.3307660>.

Singh, S., Cukier, M. and Sanders, W.H. (2003) ‘Probabilistic validation of an intrusion-tolerant replication system’, in *2003 International Conference on Dependable Systems and Networks, 2003. Proceedings. 2003 International Conference on Dependable Systems and Networks, 2003.*, San Francisco, CA, USA: IEEE, pp. 615–624. Available at: <https://doi.org/10.1109/DSN.2003.1209971>.

Sousa, P. *et al.* (2010) ‘Highly Available Intrusion-Tolerant Services with Proactive-Reactive Recovery’, *IEEE Transactions on Parallel and Distributed Systems*, 21(4), pp. 452–465. Available at: <https://doi.org/10.1109/TPDS.2009.83>.

Steinhammer, K. *et al.* (2006) ‘A Time-Triggered Ethernet (TTE) Switch’, in *Proceedings of the Design Automation & Test in Europe Conference. Proceedings of the Design Automation & Test in Europe Conference*, pp. 1–6. Available at: <https://doi.org/10.1109/DATE.2006.244145>.

Sumra, I., Ab Manan, J.-L. and Hasbullah, H. (2011) *Timing Attack in Vehicular Network*. Available at: https://www.researchgate.net/publication/260365999_Timing_Attack_in_Vehicular_Network.

Sun, J. *et al.* (2024) ‘A Tamper-Resistant Broadcasting Scheme for Secure Communication in Internet of Autonomous Vehicles’, *IEEE Transactions on Intelligent Transportation Systems*, 25(3), pp. 2837–2846. Available at: <https://doi.org/10.1109/TITS.2023.3265403>.

Vöelp, M. and Esteves-Verissimo, P. (2018) ‘Intrusion-Tolerant Autonomous Driving’, in *2018 IEEE 21st International Symposium on Real-Time Distributed Computing (ISORC)*. *2018 IEEE 21st International Symposium on Real-Time Distributed Computing (ISORC)*, pp. 130–133. Available at: <https://doi.org/10.1109/ISORC.2018.00026>.

Wei, T.Z., Razak, S.F.A. and Kamis, N.H. (2022) ‘Secure Communication for Connected Vehicles Safety Applications’, in *2022 IEEE 13th Control and System Graduate Research Colloquium (ICSGRC)*. *2022 IEEE 13th Control and System Graduate Research Colloquium (ICSGRC)*, pp. 71–76. Available at: <https://doi.org/10.1109/ICSGRC55096.2022.9845160>.

Yadav, N., Ansar, S.A. and Chaurasia, P.K. (2022) ‘Review of Attacks on Connected and Autonomous Vehicles (CAV) and their Existing Solutions’, in *2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*. *2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, pp. 1–6. Available at: <https://doi.org/10.1109/ICRITO56286.2022.9965024>.

12 Appendix 1

12.1 Types of Modelling

SAN

SAN models, also known as stochastic activity networks, have been used in performance models for decades, even being used in software such as Mobius, which is currently being used in this project to model CAV systems to investigate their cyber-resilience, so that I can see how the system behaves under different circumstances (Sanders and Meyer, 2001).

The SAN model is said to consist of the following components for its ‘Atomic’ components, namely for a single network not yet joined to any others (Sanders and Meyer, 2001):

- **Place:** These are circular objects in the network that are assigned a value to indicate a state, such as compromised or failed. The system usually transitions between these when being simulated.
- **Timed Activity:** An activity that happens in a period, set to perform a task at a fixed rate according to whatever distribution is chosen for it, such as exponential distribution.
- **Instantaneous Activity:** If an event happens on a state is reached connected to this, the activity happens immediately.
- **Input Gate:** This is a gate that the system passes through, with a predicate/requirement has based off existing states or parameters in the system which must be met to pass through this gate, otherwise the next state or event after this gate will not be reached.
- **Output Gate:** This is usually connected to another activity and values are output or set here before continuing to the other parts of the activity network.

These components are linked together via connections that can be curved or straight, with different cases occurring at activities, like if or case statements in programming. Various devices can be modelled this way, as well as being able to model Markov chains (Popov, 2015). Once the ‘Atomic’ component is created this can then be linked to other components, so that a place can exist in two different models, allowing another model, such as an attack, to affect another system. This can then be parameterized for numerous iterations and simulated either via numeric solver or Monte Carlo simulation.

STRIDE

The STRIDE model looks at six key areas for which properties being violated can result in serious security breaches, namely via attacks that compromise the following (Fowler, 2020):

- **Spoofing:** This involves assuming a false identity, such as with ECU impersonation, which means authentication has been violated so that elements such as data being communicated may no longer be legitimate.
- **Tampering:** Violating the integrity property of security by modifying data or code somewhere in the network, such as on a device.
- **Repudiation:** The security equivalent of lying, violating non-repudiation means it is possible to perform an action and deny having done so, which means actions of users on the network are hard to account for when this is an issue.

- **Information Disclosure:** A confidentiality issue caused by information being exposed to unauthorized users by various means. This is not a major concern in this project as it does not have direct implications, though what is done with the information later could have an indirect impact depending on what it is.

- **Denial-of-Service (DoS):** By depriving users of a network or system of a service, or causing a degradation in the service, resources could become limited or exhausted which violates the availability property.

- **Elevation of Privilege:** By violating authorization, an adversary that elevates their privilege can gain the ability to do things they are not allowed to do in a system, even gaining total control depending on how high they are able to elevate their privilege.

Therefore, it is important to consider these when evaluating any component in a system such as CAVs to find out what threats it is most open to and the implications of these, so that relevant security measures or modifications can be put into place to avoid any serious risks.

DREAD

This is a Microsoft developed model that provides a method to evaluate and numerically score threats that are identified in other, less numerical methods such as STRIDE modelling, looking at risk factors and rating them from 0-10, which are as follows (Fowler, 2020):

- **Damage:** This is the level of damage caused by the threat in question, which can be rated values of 0, 5, 7.5, 9 and 10. It is assumed that ten is a worst-case scenario where either the system stops working, or that its systems or important data are destroyed.

- **Reproducibility:** This is how easy it could be to reproduce the same threat, with scores going from 0 to 10 (including five, and 7.5 in between), where ten means that it is quite easy with simple steps and few tools being required.

- **Exploitability:** The level of difficulty in actually performing a threat/attack is also important, and unlike the others, this rating starts at a minimum of 2.5, with the best case being that advanced tools and knowledge are required, going up to 5, 9 and 10, where 10 is assumed to be a very easy attack that can be launched by anyone.

- **Affected Users:** This looks at the range of the effect of the attack, starting from zero where there is no effect on network users, followed by any already compromised user, some users, administrators and finally all users of the network being the worst case with a value of 10

- **Discoverability:** Aside from existence of vulnerabilities, the ability to find the vulnerability in the first place is also a key area. Starting from zero, this is the hardest, requiring access to code or configurations. A rating of five means that monitoring or manipulating requests can allow someone to figure it out, 7.5 means easy discovery of this kind of fault and ten means total visibility on the user end.

Out of all these scores, they are then summed up and divided by 5, giving the average risk level of the threat in question, though it should be noted that as these scores are qualitative (i.e. no numerical basis for their assignment) they are subjective and therefore not a solid method of quantifying the risk, particularly for more complex systems, so systems such as Common Vulnerability Scoring System (CVSS) would be advised in those cases.

PASTA

The Process for Attack Simulation and Threat Analysis consider threats from the business and technical perspectives. This approach is centred around an attacker and aims to produce output that evaluates and enumerates threat scores (Fowler, 2020).

HEAVENS (Healing Vulnerabilities to Enhance Software Security & Safety)

A systematic approach of modelling, which aims to determine the security requirements as well as performing security testing, it uses case studies, threat analysis and risk assessment, which can then be used to better identify, classify and assess threats and/or risks (Fowler, 2020).

EVITA (E-Safety Vehicle Intrusion Protected Applications)

This evaluates protection of security engineering, threats and security requirements, with the original European project having focused on protecting the in-vehicle network in the automotive domain (Fowler, 2020).

OCTAVE (Operationally Critical Threat, Asset and Vulnerability Evaluation)

This form of assessment looks at information security, identifying the threat, the assets and evaluating the vulnerability, so that the assets most important to an organization can be better protected by seeing which threats or vulnerabilities put these assets in danger (Fowler, 2020).

CVSS (Common Vulnerability Scoring System)

In CVSS, vulnerabilities that are identified are then given scores between 0-10 (10 being the most serious), with different attributes of the vulnerabilities being calculated as follows (Fowler, 2020):

- Attack vector, namely a physical, short-range attack or a remote attack.
- Complexity of the attack, which means how difficult it can be to perform, including the type of steps needed to perform it.
- Level of privilege required to perform the attack, meaning how much access is needed.
- Level of interaction needed with the system for an attack, as well as how much further an attack can progress (i.e., to do other attacks). How much confidentiality, integrity and availability are affected by an attack. The individual values are then combined to determine a final score from 0-10 that determines the severity of an attack. The CVSS is considered a standard method of ranking security issues, and is considered quantitative (Fowler, 2020).

FMEA

FMEA, known as Failure Mode and Effect Analysis is used to identify potential problems and their impacts. The focus here is mostly faults as opposed to direct security issues, and it is usually organized as a spreadsheet consisting of the following elements (Forrest, 2010):

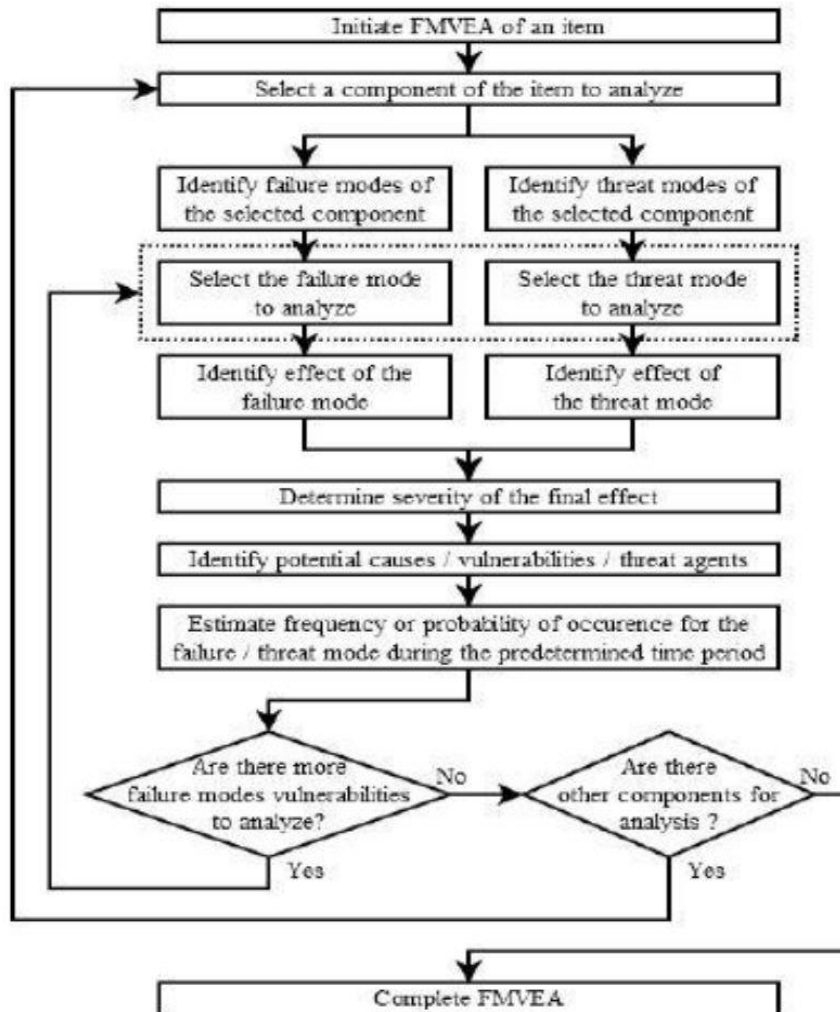
1. What is the function or process in question?
2. What is the type of failure that has occurred?
3. What are the potential impacts of this failure?

4. What is the severity of such a failure?
5. What are the potential causes of this failure?
6. How frequently does this failure occur?
7. Existing controls or solutions that allow prevention and detection of such a failure.
8. How difficult is it to detect this kind of failure?
9. What is the risk level of such a failure?
10. What actions are recommended to deal with this?
11. Who is responsible for these recommended actions?
12. When should this action be taken?
13. What actions were taken?

At this point the severity, frequency, detection, and risk level should be calculated again. When performing FMEA, it is important to look at how severe the effects are on the customer/user, as well as the frequency at which it may occur and how easily it can be detected when it does occur. Scoring for each value such as frequency, detection and severity should be agreed on in advance for each failure mode, ideally with prior data of such occurrences to justify the qualitative values decided upon (Forrest, 2010).

FMVEA

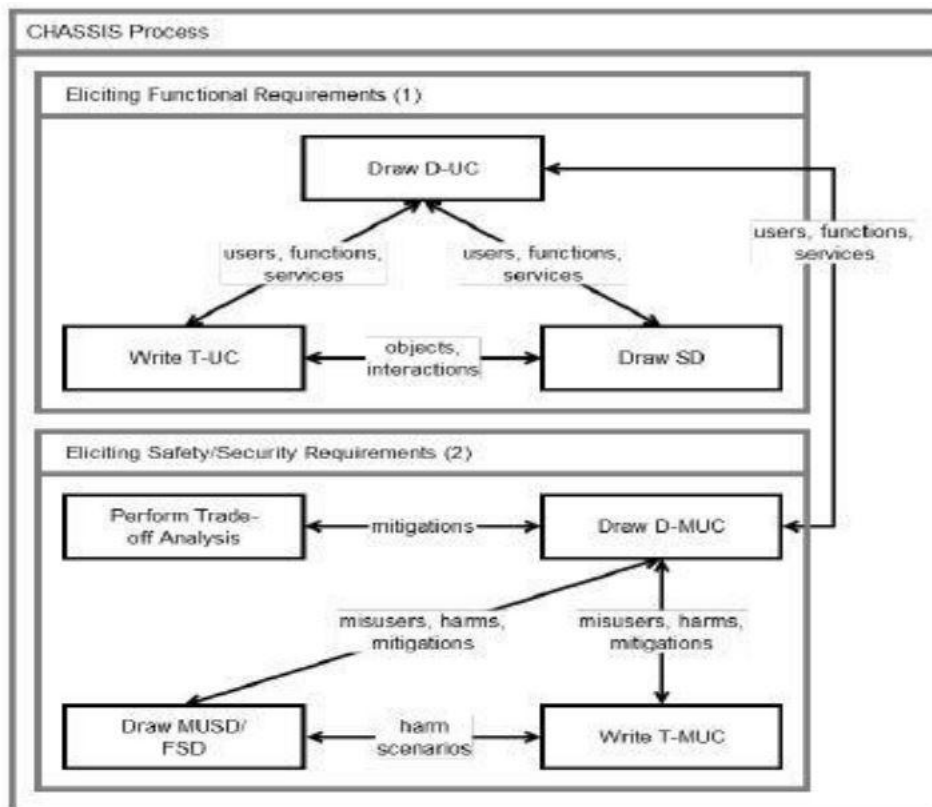
Much like FMEA, with the difference of looking at vulnerabilities as well, hence the V in FMVEA, meaning that as opposed to safety, this is suitable for safety and security at the same time, as it considers the effects of vulnerabilities that could be exploited as well (Schmittner, Ma and Smith, 2014).



The above flow chart (Schmittner, Ma and Smith, 2014) shows the full method for FMVEA, starting by choosing a component, for which identification, analysis and effects of both failure and threat modes are done in parallel. Then, they are combined to determine severity, followed by cause and the frequency/probability of their occurrences (Schmittner *et al.*, 2015). From this, I check if there are more modes to analyse, and if so, I return to the analysis stage, otherwise I check for other components to perform FMVEA on, which will determine if I start the process for a new component or if this is the end of the FMVEA process. Failure modes are used to describe the ways in which the system fails, while the threat modes are based on threats determined in advance using a model like STRIDE (Schmittner *et al.*, 2015).

CHASSIS (Combined Harm Assessment of Safety and Security for Information Systems)

CHASSIS looks at use cases and diagrams of sequences that look at both the functional safety and security requirements of systems. The diagrams aim to describe all the functions, services and users involved (Schmittner *et al.*, 2015).



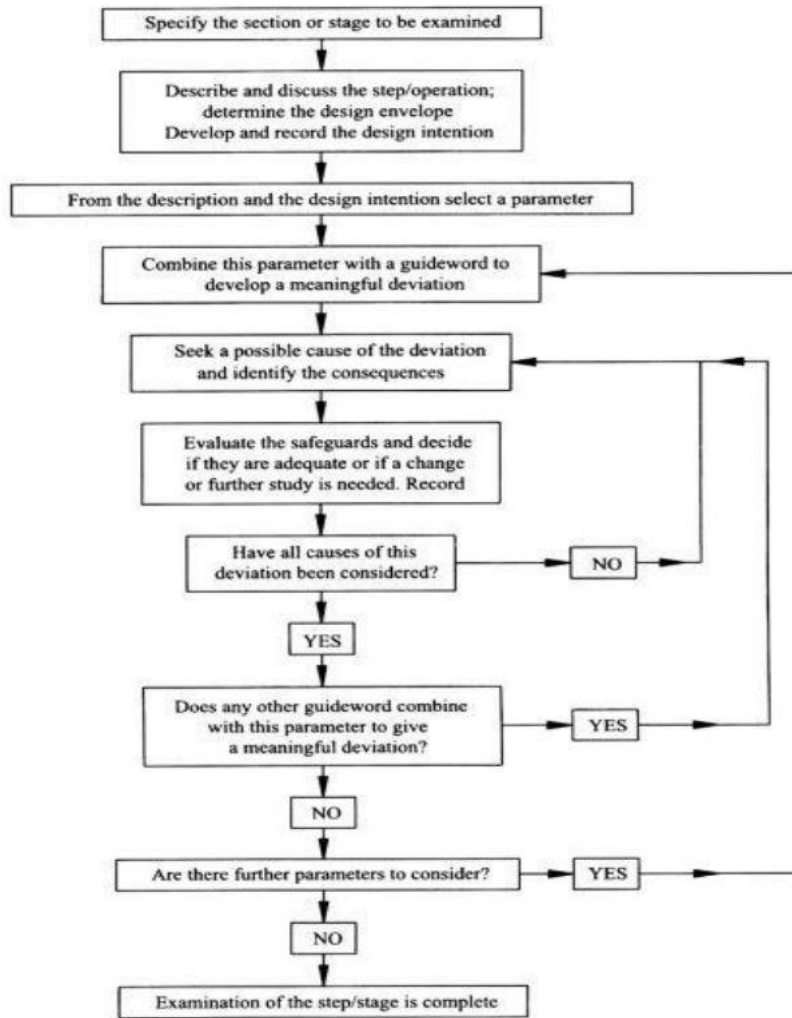
Above I can see the CHASSIS process (Schmittner *et al.*, 2015), consisting of two sections, the functional requirements and the safety/security requirements. These and their internal workings can be described as the following (Schmittner *et al.*, 2015):

- **Functional Requirements:** Using data on users, functions and services, the use case diagram (D-UC) is created. The data on users, functions and services is then used to create a textual (written) use case as well as another sequence diagram. These also both exchange information on objects and interactions, creating more refined use cases and diagrams.
- **Safety/Security Requirements:** With the data on users, functions and services, misuse case diagrams (D-MUC, which can show cases where the system is abused) are created, with use cases obtained through HAZOP. From this, mitigations are looked at with an analysis of the trade-offs of implementing them. Then, the textual version of the misuse case is created, with data on misusers (i.e. those doing the exploiting), potential harms/damages and mitigation methods. Finally, these combined with the scenarios in which harm is done are used to create what are known as misuse or failure sequence diagrams (MUSD/FSD respectively).

From this, I now have the use cases, those using the system, interactions, as well as threats, potential mitigation and scenarios all noted down so that they can be compared and combined where needed to see safety/security issues and where they map onto the function of the system or components this is being modelled for (Schmittner *et al.*, 2015).

HAZOP (Hazard and Operability Study)

HAZOP is designed as a method of analysis for systems or processes where the information on the system design is very detailed (i.e. in schematics) and is usually carried out in stages by a large team that possess knowledge of multiple disciplines (Crawley and Tyler, 2015).



Like FMEA, it is important to identify causes, evaluate the potential consequences, as well as looking at safeguards/defences and recommended actions. In the above diagram (Crawley and Tyler, 2015) the process is shown for HAZOP when looking at parameters, where a stage or section of the system/process is selected, and the parameters are looked at one by one to identify deviation (i.e. unexpected or undesirable behaviour), and keep doing this until all have been identified for all parameters (Crawley and Tyler, 2015). If done properly, all hazards, their implications as well as potential countermeasures or improvements will be clear before designing and fully implementing any system.

12.2 Background & Literature Review (Extended)

Background & Literature Review

In this section I looked at the background of CAVs and conducted a review of the relevant literature through the method of systematic literature reviewing, as specified by (Kitchenham and Brereton, 2013), by first finding a sufficient number of relevant papers for study through manual searches, selecting papers and excluding those that did not pertain directly to CAVs, AVs or their safety and security. The references of these papers were then also checked for more relevant sources, as I collected more sources on the various parts of CAV relevant to my subject. These included machine learning, the in-vehicle network, vulnerabilities, defences, and other components (i.e., vehicular safety or communications). While various sources of literature were used, the main ones consisted of IEEEExplore, arXiv and ScienceDirect.

In-Vehicle Network

This section addresses the components of the network inside the vehicle, used to communicate data between different subsystems which are used to perform different tasks, as well as different protocols and connections used to do this.

Electronic Control Units

Increasingly new features are being integrated into the vehicle. These features need to be controlled properly in an electronic system, so embedded systems called ECUs were introduced in order to make the tasks easier (Alam et al., 2019). Each ECU takes sensor inputs which are processed to get data for the relevant ECU to perform its task. This is not always direct, so one ECU could have the necessary data and communicate it to another ECU where necessary. Every ECU is a small computer connected to a sensor as well as other ECUs, and an actuator through which the software decisions are converted to physical actions where needed.

Data is usually communicated through the in-vehicle buses which link all the ECUs, with examples consisting of CAN, LIN, FlexRay and more. There may not always be security implemented on these buses, so ECU data should be kept secure by various means such as encryption when it is communicated, or to make sure there is some form of access control, as the current method of all ECUs being able to communicate with one another freely could lead to the exploiting of the entire system through a single ECU (Alam et al., 2019).

There are numerous ECUs in vehicles, many of which can be used for a range of tasks, including but not limited to the braking, steering and transmission (Aliwa et al., 2020). The data collected by ECUs is critical to their actions and decisions, such as those obtained by the brake module, which needs data such as speed from sensors to know when to brake. The engine control looks at temperature, fuel and more in order to know how the engine should behave (i.e. if the engine is hot, it should do less work), as well as other modules such as the tyre pressure monitor system (TPMS) which gives an idea of what kind of speed to travel at based on the pressure (i.e. remaining air) (Pham and Xiong, 2020). Therefore, any incorrect data will lead to incorrectly made decisions regarding action taken by the vehicle and its ECUs.

The ECUs, specifically some of those mentioned can be used to control safety-critical functions, meaning that they must not be compromised under any circumstances if the vehicle is to remain safe. As the functionality and the communicated data of the ECU can both have effects on the performance of the vehicle, it is vital that the ECU and its data both remain secured. The ECU must always remain available with the integrity of its data being preserved.

ECUs are what controls the different systems in vehicles as well as their functions, which is an important part of the in-vehicle network for which security was explored by (Alam *et al.*, 2019). It is stated that once an ECU is accessed it can send messages to all other ECUs to exploit them as well, as well as modifying ECU data. The authors here proposed that cryptographic methods be implemented on the ECUs, as well as identity-based access to control what each ECU is allowed to do in terms of communications, with a “watcher” designed to monitor data in case it is modified (i.e., by an adversary). All ECUs communicate with a “Mother” or master ECU in between them checking their communications to ensure

that ECUs are legitimate, approving communications using a list of permitted ECUs, as well as passing the encryption and decryption keys between the ECUs. There are also multiple “Mother” ECUs which can then check by consensus to approve any communication or alterations made by one of them in case it has been compromised. Experiments were carried out on Raspberry Pi boards with the relevant software for ECUs loaded onto them. Experiments were designed to check different properties such as integrity/authenticity, access control, and data security. The authors also found the following maximum end-to-end delay values (Alam *et al.*, 2019):

1. Control data: 10ms
2. Real-time audio/video: 45ms
3. Multimedia data: 100ms

Therefore, as control data requires the lowest time this should be the maximum requirement of time for any single transmission, including encryption, decryption, verification, and transfer. The experiments conducted showed that more than 16kB of data can be transmitted effectively in 10ms, but the authors did not have sufficient data for comparison to vehicular requirement, which could be done in future works (Alam *et al.*, 2019). A limitation was that the ECU was using a single-board computer, but the authors believe this should change as vehicles become more advanced, as they will also need more powerful ECUs.

Controller Area Network (CAN)

The CAN bus was invented in the 80s by Bosch as an inexpensive means of connecting nodes (i.e. ECUs) in the vehicle along a single shared line, where 0s are considered to be dominant (i.e. higher priority bits) while 1s are considered recessive (Sharma and Möller, 2018). This system of bits is used for bus access, using CSMA/CA (Carrier Sensitive Multiple Access with Collision Avoidance), where more zeros in the ID section of the CAN communication frame gives it higher priority, so that higher priority frames can communicate in real-time with less traffic.

This form of communication, also known as bit arbitration, is an efficient standard for in-vehicle communication, as it is possible to assign more zeros to the more important frames, such as those which involve safety-critical systems. There is no central host in the CAN bus, with different ECUs and components being able to easily communicate with one another over the CAN bus, which is currently considered the default in-vehicle bus.

The key capabilities of the CAN bus consist of the following (Forsberg and Hedberg, 2009):

- Any node/ECU should be able to transmit messages across the bus if it is idle.
- The worst-case message delays should be possible to calculate to determine the guaranteed latency.
- It must be possible to add or remove devices from the bus without changes having to be made to the hardware or software.
- A priority system must be in place so that high priority messages get access before lower priority ones.
- Any messages transmitted on the CAN bus should be received by any node connected to the CAN (i.e., broadcast to all).

- There must be error detection and retransmission capability of all messages, with distinction between the types of errors (i.e., temporary and those caused by a defective node).
- There must be arbitration to determine which node sends first by checking their priority if two nodes intend to transmit at the same time.

The general structure of CAN frames is shown in Figure 1, consisting of the frame start, header, control field, payload, trailer (i.e., CRC for error check) and the end of file. This can then be broken down further to specify the individual bits of a standard frame or extended frame.

The CAN protocols are as follows (Aliwa et al., 2020):

- High Speed CAN (CAN-H) has bitrates between 125Kbps-1Mbps, being used in real-time applications that are critical such as the engine and brakes, mostly in the powertrain and chassis domains.
- Low Speed CAN (CAN-L) has bitrates from 5Kbps-125Kbps, being used mostly in the body domain where functions are non-critical (i.e., doors and windows).
- CAN with Flexible Data rate (CAN-FD) can go up to 10Mbps, being used for critical applications that work in real time, much like CAN-H, in the powertrain and chassis domains.

The frames in the CAN bus protocols can be grouped into the following four (Aliwa et al., 2020):

- Data frames, which carry data payload, such as information from sensors.
- Remote frames, which are used to request transmission of data from another ECU. There is no data field in this frame.
- Error frames are used to indicate that there is some sort of error in the bus.
- Overload frames are used by an ECU to state that it is too busy to receive a message from another ECU, so it will have to wait.

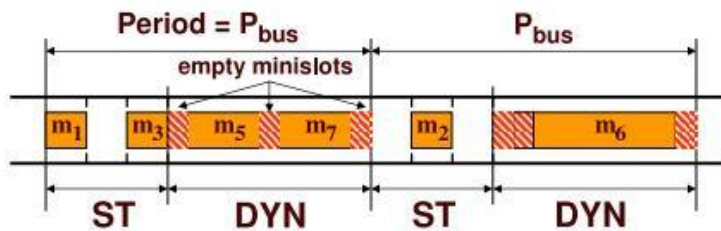
The original CAN bus was created with no consideration that there could be outside communication and therefore any access to it by any form of attack, meaning that security was never considered. Therefore, this is an area of the vehicle that contains many vulnerabilities that will need to be evaluated, especially as there are other in-vehicle buses in existence that could offer better options, such as FlexRay.

The authors of (Aliwa *et al.*, 2020) explore the in-vehicle network, mostly the bus protocols (i.e. CAN, FlexRay and LIN), vulnerabilities and defences. The CAN bus and in some cases FlexRay are assumed to be the default in-vehicle buses for the critical parts of the in-vehicle network. It is stated that the CAN bus did not initially consider security and consists of low or high-speed CAN buses depending on how critical communications are. Modules such as the telematics unit and radio are connected to both buses which means they could be used to access critical systems in an attack. The CAN itself is vulnerable to attacks such as DoS, impersonation (i.e., fake ECUs) and false messages which should be modelled as they can have far reaching effects that could even allow control of the in-vehicle network. Proposed mitigation methods include cryptography and IDS, which could be used to detect the attacker. These are limited by the amount of overhead, cost and data required to implement them, as

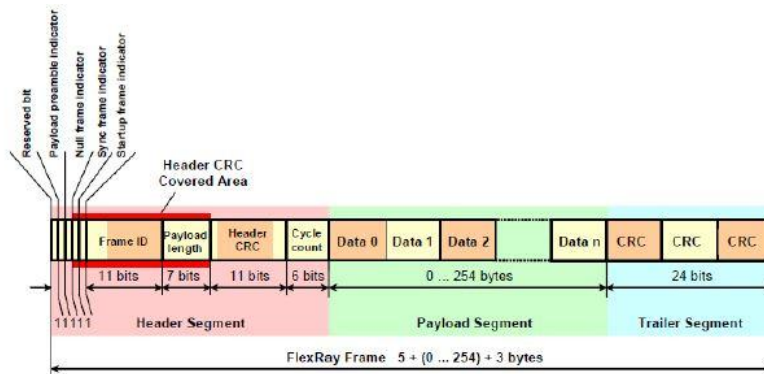
well as changes that may need to be made to the in-vehicle bus. The CAN should be considered the default in-vehicle bus, and variations on CAN such as CAN-FD or CAN-XL could hold the solutions such as the added security needed to improve the CAN bus security.

FlexRay

Where there are high bitrate requirements in the in-vehicle network, the in-vehicle bus could also be FlexRay, which offers up to 10Mbps data rate, with fault tolerance support as well as support for high speed, critical applications (i.e. X-by-wire) (Aliwa et al., 2020). Like CAN-H and CAN-FD, it works on the powertrain and chassis domains for ECUs.



The FlexRay frame, as shown above consists of a static (ST) and dynamic (DYN) segment, which are divided into equally sized slots that are then divided further down into minislots (Hagiescu et al., 2007). Each slot or minislot is assigned to a task, with the DYN segment tasks being in order of priority, while the ST segment is allocated to tasks from the start, and they will only ever send during that slot. If there is nothing to send, the slot will go empty, as other tasks may not use it. The exception to this occurs only in the DYN segment, where a task may only send in the DYN segment if its size fits into however much space of the segment is left at the time of transmission.



In the above diagram, the FlexRay frame is shown to consist of a header, payload, and trailer. These can be broken down into the following (Forsberg and Hedberg, 2009):

- The header consists of information such as synchronization of the frame, ID, payload length, and CRC check for the header to make sure no accidental changes to the data have happened.
- The payload segment, containing up to 254 bytes of data to be communicated to another node in the in-vehicle network.
- The trailer segment, which is twenty-four bits of error checking calculated across both the header and payload segments.

Due to different segments and therefore different access protocols (static and dynamic) for each, FlexRay is found to be more complex, but it is considered more secure. This is because

unlike the CAN bus, which has a single channel as opposed to the two-channel structure of FlexRay. The additional channel can be used as a backup if one fails whereas if the CAN bus fails many nodes will instantly be cut off from one another (Forsberg and Hedberg, 2009). This speaks well for maintaining the availability of the system and therefore reducing safety implications of a channel failure. As its name indicates, FlexRay is supposed to be flexible, and does so by allowing existence of various topologies in design, unlike CAN, which can only be a line/bus topology (Forsberg and Hedberg, 2009).

Authors of (Forsberg and Hedberg, 2009) here looked at FlexRay and CAN bus, two in-vehicle buses to compare in terms of their effectiveness in communication. Vulnerabilities were not considered in this evaluation so this would also need to be considered if evaluating resilience. In terms of complexity, CAN is low complexity unlike FlexRay which uses two different protocols for accessing the bus (static and dynamic segments) which increases the complexity. In terms of safety, CAN has low complexity meaning less risk of errors, but the line topology of CAN means a break could result in many ECUs being unable to communicate, while FlexRay has redundant channels to maintain function if one is to fail. CAN is quite flexible but not as much as FlexRay, hence the name due to how it can be configured in many ways as well as the ability to include various topologies. CAN is the current default bus, and it will take a while before FlexRay is to really be put into use in vehicles.

Following on from comparing the buses, the performance of FlexRay as a network for ECUs is looked at by (Hagiescu *et al.*, 2007). FlexRay is modelled as a frame structure consisting of the static and dynamic segments. A model was created and then implemented in MATLAB, showing around 393ms delay from the sensor to the actuator in getting the task done. Modelling of FlexRay and other buses could be an area worth exploring further, especially as the authors stated their interest in exploring the idea of integrating such models into tools used to design FlexRay systems in future.

Local Interconnect Network (LIN)

Used in vehicular components that are low cost as well as having low bitrate requirements, the LIN protocol is used for ECUs that have no critical function, such as doors, lights and A/C, as there is no need for a stable robust network for these components or the domain they may be placed in (Aliwa et al., 2020).

The bitrate of the LIN can range from 1-20Kbps, with a single master that can support up to 16 ECUs, all connected along a single wire, mainly being used in the body domain of the vehicle (Aliwa et al., 2020). LIN is usually used within in-vehicle networks that also contain the CAN bus, but due to their differences in architecture, CAN vulnerabilities do not always necessarily apply to the LIN (El-Rewini et al., 2020).

The LIN has a master node which communicates with multiple nodes that take instructions from it, known as slave nodes. The schedule of communication is time based so the master knows at exactly what time to send messages. The master sends a header, intrusting the slaves/ECUs to subscribe or publish to the bus (i.e. send/receive) (El-Rewini et al., 2020).

Media Oriented Systems Transport (MOST)

MOST is an in-vehicle bus system used mostly for entertainment, where data such as audio and video can be transmitted using fibre-optic cables (El-Rewini et al., 2020). The MOST consists of a master node that sends out frames to make sure all the nodes are operating at the same time (i.e. are synchronized), with transmissions containing static and dynamic segments, much like FlexRay, as well as a control channel (El-Rewini et al., 2020).

MOST consists of three types; MOST25, MOST50 and MOST150, with higher bandwidth provided as the version/number of the MOST increases (e.g., MOST25 gives 25Mbps, MOST150 can give up to 150Mbps). Each frame contains a control channel, a streaming data channel and a packet data channel (Grzemba, 2011). The vulnerabilities must still be explored, such as the possibility of disrupting synchronization, which could put the nodes out of phase with one another so they are no longer all operating at once (El-Rewini et al., 2020).

Another form of bus is described by (Grzemba, 2011), known as MOST which has types 25 (i.e. MOST25), 50 and 150 which are their bitrates in Mbps. MOST is considered a backbone for the infotainment system which aims to be function-oriented and provide transmission of streaming, packet data, and control information. MOST follows the OSI model, with both synchronous and asynchronous data transfer modes being available. The layers for MOST can be divided into the following:

1. Electrical or optical physical layer MOST, which happens at the physical layer.
2. Network interface controller for MOST is at the data link layer.
3. For network, transport and session there is network service as well as part of the streaming services.
4. The presentation layer contains part of the streaming service and network service.
5. The rest of the stream service as well as the function blocks are in the application layer.

The MOST network can be a ring network of all the nodes, with a master and many slaves, where all are under the command of the master, with all nodes having different connections, the outermost being an interface between the human user and the machine.

Automotive Ethernet

Ethernet as an in-vehicle bus system is a newer idea, intended for future use, consisting of multiple hosts connected by switches, which act as bridges for sending frames to other nodes in the network, with the use of IP protocol owing to its network-based structure like LANs, with the nodes consisting of ECUs.

Relevant protocols for automotive Ethernet were found in (El-Rewini et al., 2020) to be time-triggered and time-sensitive, with time-sensitive synchronizing frames so that they can be grouped according to priority, therefore having a priority system like CAN, while time-triggered allows the support of more than just one type of traffic. The time-triggered traffic consists of (El-Rewini et al., 2020):

- Time-triggered (TT) traffic, which allows transmission at specific times and requires clock synchronization.
- Rate-constrained (RC) traffic, which sets the bandwidth in advance but has a lower efficiency than time-triggered.

Time-Triggered Ethernet

The authors of (Steinhammer *et al.*, 2006) propose the use of a time triggered ethernet switch. This form of ethernet combines both event and time triggered into a single communication system, with the switch itself allowing for predictable time triggered ethernet while maintaining the support for the old ethernet traffic. A standard ethernet switch was found to be insufficient while a time triggered switch works better, remaining conflict free in the way data is transmitted.

Time triggered ethernet provides the ability to transmit both event-triggered and time-triggered traffic using a single system. This means the event triggered will not have strict times while the time-triggered will have fixed times with a precalculated amount of delay (Nikishin and Konnov, 2020).

The weaknesses of time triggered technology are as follows (Nikishin and Konnov, 2020):

1. Resources wasted on calculating the time-triggered delays each time.
2. System defences being deployed will interrupt the communications and reduce bandwidth.
3. The conflicting time frames and low bandwidth could lead to poor control of the dispatch time.
4. Timeslots could be restricted so transmission becomes harder if you cannot send multiple frames in the same timeslot to different ports.

In (Liu *et al.*, 2019) it is found out that the current state of ethernet has been primarily event triggered. However, this type of transmission has a delay which accumulates per each hop which can result in increased delays each time as the number of hops is also increasing. Therefore, the use of time triggered ethernet has been proposed to ensure a fixed delay, which is compatible with both best effort and rate constrained policies for networks that already exist. The event triggered information is sent in the slots where the time triggered data is not transmitted, allowing both to be transmitted in combinations (i.e. by sharing the same communication and channel). It was proposed that based on the percentages of the bandwidth used for event and time triggered, it can be scheduled in advance how to transmit both sets of data.

The authors of (Kopetz *et al.*, 2005) tell us that time-triggered ethernet can support a range of applications including multimedia and safety critical systems in real time. This paper discussed the idea of using time triggered to unite the time and event triggered into a single coherent form of communication that allows for both to be used at once. This form of communication works for both regular and real time, making it suitable for CAVs and other X-By-Wire systems.

Requirements of a time-triggered protocol include (Kopetz *et al.*, 2005):

1. Predictable message transfer
2. Fault tolerant with global time base
3. Fault isolation

4. Consistent diagnosis

5. Scalability

Event triggered and time triggered are both activated by significant events, non-temporal, and temporal, respectively. Event triggered can be an event inside or outside while time triggered tends to focus on significant events inside the system. The significant temporal event in time-triggered is basically when the inside of the system reaches a predefined moment in time also known as the trigger value. Event triggered messages are the even messages while time-triggered is known as state messages. It was proposed that systems such as scheduling could be used to improve the design of time-triggered and to include fault tolerance to create a better version of the existing protocol (Kopetz *et al.*, 2005).

(Loveless *et al.*, 2023) discovered the possibility of attacking time-triggered ethernet protocols (PCSPOOF), which can in turn have serious safety implications due to the application of time-triggered ethernet in real-time applications, some of which include safety critical systems. The exploit involves using the idea that best-effort (i.e. non-critical devices also using the same network but separate from the time-triggered components) can be used to determine information about the time-triggered area to craft malicious messages (i.e. synchronization messages). It is also possible to inject electrical noise into time-triggered switches to allow best effort devices to trick them into sending the malicious messages to other devices so they will lose synchronization which results in messages being dropped meaning failures of safety critical systems in vehicles and aircraft. The best-effort devices are typically ignored due to being isolated and can be from manufacturers who do not secure them properly, usually not being regarded as a concern in any shape or form.

Proposed mitigations included (Loveless *et al.*, 2023):

1. Blocking electromagnetic interference to avoid this sending of electrical noise.
2. Authentication on the link layer.
3. Hiding of key information such as data fields so that information cannot be inferred from them easily.
4. Check sources addresses and preamble lengths to ensure legitimate frames are being sent.

From this it can be seen that time-triggered ethernet is not as secure as originally thought, so some measures must be considered regarding components previously considered irrelevant to it, as the mere fact that they share the same cabling threatens this isolation guarantee, posing a threat to safety-critical systems that make use of time-triggered ethernet and maybe even ethernet in general in their networks.

Sensors

Sensors are one of the most vital elements to CAVs, acting as the starting point of perception for the vehicle. Without these, there would be no information on the surroundings for the vehicle to process, meaning that it would be making decisions while blind to its environment. Here I look at the key sensors and their functions. Sensors are known to work across different ranges due to their limitations and can be described as the following (Qayyum *et al.*, 2020):

- Proximity sensors, which have 5m range, are typically ultrasonic sensors used to detect obstacles at close range, particularly when the vehicle is moving very slowly. These types of sensors can help in tasks such as parking, as they can make sure that a safe distance is kept from nearby obstacles when adjusting the vehicle position.
- Short-range sensors have 30m range, being used in forward cameras, backward cameras as well as in short range radar (SRR). Forward cameras can identify upcoming objects ahead, while backwards cameras work similarly for objects behind. Blind spot detection is also a notable application of SRR.
- Medium range sensors operate from 80-160m, consisting of LiDAR or medium range radars (MRRs) that can be used for detection and collision avoidance.
- Long-range sensors can detect obstacles up to 250m ahead, working at high speeds and collecting additional data from surrounding devices or communications to allow for cruise control, where the driver does not need to focus as much.

In this section the focus will be on LiDAR, RADAR, GPS, and cameras, as these are the best known and more common sensors for CAVs, with some mention of the additional sources of sensor data which help to improve the overall reliability of environmental perception data gathered by the vehicle.

Light Detection and Ranging (LiDAR)

LiDAR is a type of sensor which uses laser light to get in-depth, 3D data points regarding different objects in the environment surround a vehicle. By creating a 3D map of the surroundings, LiDAR allows the vehicle to perceive and navigate its surroundings in a safer manner, given that it is also considered a more reliable class of sensor with accurate object detection (Cao et al., 2022). The wavelengths of LiDAR range from 10 μ m-250nm, meaning it can go from infrared to ultraviolet light (Pham and Xiong, 2020).

The pulses of light sent from the LiDAR will hit nearby objects and be reflected back to be received by the sensor, using the time taken to return in order to determine the distance between the sensor and nearby objects (Parkinson et al., 2017). LiDAR can provide 360 degree view (all-round) with over 16 channels, as well as the rotation mechanisms being classifiable as mechanical, semisolid-state and solid state (Dibaei et al., 2019). If the signals can be prevented from reaching the sensor or tampered with, this could be an area of critical concern as the ability of the vehicle to perceive the environment has now been adversely affected.

Radio Detection and Ranging (RADAR)

Using EM waves in the radio or microwave range, it is possible to apply the same idea as LiDAR by calculating the distance based on time taken for a reflected waveform to reach the radar sensor, which can serve various purposes such as helping with lane keeping and distance control (Pham and Xiong, 2020). It is also possible to use millimetre-wave radar, which can penetrate smoke and dust, where other wavelengths may not be able to, but the resolution and accuracy of the radar can still be weak, meaning that it should ideally be combined with other sensors to get a more accurate overall picture of the surroundings (Dibaei et al., 2019).

Camera

The camera is an image sensor for the vehicle and can be used for object detection, possibly as a cheaper alternative to LiDAR, but can perform badly in certain weather (i.e. rain and fog) (Pham and Xiong, 2020). Due to this, they are usually paired with other sensors, with their main purpose being identifications of things such as traffic signs, and are either mono or stereo vision, with the difference being that more cameras are needed in mono than in stereo (Parkinson et al., 2017). As the sensor is strictly image based, much like the eye, the quality can depend on the weather or visibility.

Global Positioning Systems (GPS)

This sensor is for navigation, being based off a series of satellites in space owned by the US government which transmit high frequency signals that can be detected by any device on Earth which has the ability to receive GPS signals, such as smartphones or vehicles that have GPS receivers built into them (Pham and Xiong, 2020). The receiver in any device is able to use the signals from the GPS satellites to compute its present location to within 1m of accuracy, but in the case of obstructions that could weaken the signal from a satellite, more satellites will make it possible to continue to get an accurate position (Parkinson et al., 2017). There is no means of verifying if the GPS signal is authentic, as the encrypted signals are for the military only, meaning that it could be possible to compromise the integrity of the GPS receiver.

Architecture

A fundamental part of the CAV structure is its architecture, as this is how different components of the system will be joined together and communicate. It is important to have an efficient architecture with good security as well as allocation of resources such as bandwidth.

The traditional in-vehicle architecture in Figure 2 is *gateway-based*, consisting of ECUs all connected to the gateway, each with the necessary sensors, actuators or other sub-bus components. However, as the default in-vehicle bus is assumed to be CAN, the CAN bandwidth needs to be shared between all ECUs (Dibaei et al., 2019), meaning that due to the lack of discrimination in bandwidth allocation, high-priority communications by key ECUs could be delayed if the bandwidth of the bus is already being used up by others, suggesting that a better architecture is needed.

The proposed alternative in Figure 3 is *domain-based*, with ECUs being grouped according to similar functionality, each under a relevant domain controller, so the bandwidth is allocated to each domain in advance. Their power is linked along the electric grid of the in-vehicle network, as well as a network backbone for the allocation of bandwidth and power (Dibaei et al., 2019). Therefore, each domain now has more bandwidth and power allocated which allows for more complex vehicle functions. Additionally, the sensors can be connected to the domain controller ECU so that sharing bandwidth with the other ECUs in the domain is no longer an issue (i.e. communicating sensor data between ECUs in the same domain), and data communication inside the domain will not affect the overall bandwidth of the in-vehicle network or its backbone (Dibaei et al., 2019).

In more detailed domain-based architectures, CAN, LIN, and FlexRay are assumed to be the more popular bus protocols the vehicle may use. In this example, the following domains were specified (Aliwa et al., 2020):

- **Powertrain:** Critical functions such as the engine and transmission ECUs, which need real-time response and will therefore use a high-speed or flexible-data rate CAN (CAN-FD).
- **Chassis:** This consists of ECU such as brakes, suspension, and steering. These provide real-time and safety critical functions to the vehicle, so the CAN will once again be one that can provide high speeds/data rates, or even the FlexRay protocol instead.
- **Body:** These are not particularly critical systems, as the ECUs will deal with things like lights, the dashboard and windscreen wipers. This domain either uses low speed CAN or the LIN as its bus.
- **Telematics & Infotainment:** Mostly used to manage communications, data and entertainment in the vehicle, no bus is specified, but it has access to the gateway that links to the other domains, making it a potential security risk if compromised.

In addition to these, the different sensors such as light sensors and cameras are also linked to the vehicle through the gateway, as their data could be needed by the different domains to take the correct actions depending on their function. Therefore, any potential weaknesses and threats to the sensor system should also be considered when evaluating any security.

Another architecture proposed in Figure 4 suggests a *central computer* instead of mini computers for each ECU or domain, so all tasks and functions exist in the computer domain, which has access to all the sensors and actuators at once (Dibaei et al., 2019). This allows for total sensor fusion (i.e., combined data from all sensors at once to visualize the environment), but the demands on data communications will increase as a result, as well as adding the possibility of a single point of failure or compromise for an attack on the in-vehicle network. The different parts of the network are divided into sub-network components according to their position in the vehicle or network properties to make them more efficient and are each controlled by a zone controller (Dibaei et al., 2019).

While these vehicular architectures have all shown to have different strengths and weaknesses in general, their security is yet to be rigorously evaluated in the form of models, which could be used to find out which architecture has the best cyber-resilience.

Safety

In vehicles, safety is a key element, as car crashes are already a known issue that exist due to human beings, so CAVs must aim to reduce the number of crashes, therefore meaning that safety measures should be implemented to reduce the rate of accidents in autonomous driving compared to regular driving, as it boils down to the statement by (Koopman, 2022) which asks what level of safety is truly safe enough. Specifically, the components that deal with or could affect safety when not functioning properly are known as the safety critical systems, with models on security taking these systems into account and evaluating their safety with or without attacks, ideally as models (Popov, 2015).

Some AV vendors assert that their products already demonstrate better safety than the vehicles driven by human drivers. Waymo, one of the leading AV manufacturers in the USA claim that the accidents with their robotaxis are less severe than the accidents with human drivers (Favaro *et al.*, 2023).

One of the issues with CAVs is that safety guarantees cannot easily be made, meaning that unless CAVs are able to promise and deliver on improved safety, as well as sufficient security to maintain the safety both in the event of a fault or compromised system, it is unlikely that drivers will have any interest in switching to autonomous driving (Shalev-Shwartz, Shammah and Shashua, 2018). In addition to this, any safety solutions implemented must be scalable so that if there are millions of CAVs on the road the solution can be scaled up to suit all of them without incurring massive costs or becoming exceedingly complex.

To achieve better safety, the Responsibility Sensitive Safety (RSS) module is written as a policy consisting of the parameters of a vehicle such as its speed, distance, and acceleration to ensure that the vehicle is always behaving safely. The RSS essentially works as a mathematical model to describe what a safe driver is to the CAV, working as a set of rules that it must follow, which can be broken down into the following (Shalev-Shwartz, Shammah and Shashua, 2018):

- No hitting road users from behind.
- No reckless driving (i.e., sudden cutting into lanes).
- Wait for the right of way to be given on roads, do not try to take it.
- Be extra cautious where there is limited visibility.
- Avoid causing an accident wherever possible provided it can be done without causing another one.

Obviously, these rules are implemented as a mathematical formula that makes use of the relevant subsystems such as sensors and ECUs. The parameters would need to be well chosen, as having very relaxed parameters could reduce overall safety by having weaker reactions, while a conservative policy (i.e. too much safety) could result in a very impractical and slow driving experience (Oboril and Scholl, 2021).

For a constant acceleration, the minimum distance can be modelled using the velocity of the vehicles ahead and behind the driver, as well as the maximum possible acceleration and deceleration, so that if the distance is less than this value it is considered unsafe (Oboril and Scholl, 2021). It is important to have a good deceleration as quick, hard deceleration could result in being rear-ended while slow deceleration could result in an unsafe distance or a collision with the front vehicle, meaning that optimal values should be determined.

Therefore, the safety of the vehicle is based off the safety-critical ECUs such as those controlling braking and steering, the in-vehicle bus as and the RSS that governs the safety. These features as well as any data gained from the surroundings by methods such as sensors, communications and how they are processed by the machine learning model will be important in modelling when considering safety and security together.

The authors of (Li, Wang and Zhang, 2021) propose an attack method against cyber-physical systems such as CAVs, known as Chronos. This attack aims to disrupt synchronization and add delays to the perception system, which can in turn affect planning and control as the sensor data communicated is not all synchronized, so the fusion will no longer work as effectively as usual. Attacks on the sensor fusion will interfere with the timing between the sensors, while delays added to control will cause deviations in the results compared to what they really are over time. Attacks were conducted on various levels, against both drones and vehicles which were simulated on computers with simulators and datasets. The experiments

showed that the attack caused serious deviations sometimes completely outside the intended trajectory which could result in serious accidents. Authors recommended more studies and modelling of this attack to better mitigate it in future as it can post major risks to cyber-physical systems.

It is noted that in (Lisova, Sljivo and Causevic, 2019) the authors have found 33 sources of literature and have indicated that safety and security co-analysis is a growing field in the area of engineering. Several papers were presented by the authors in their reviews, which indicated that approaches to safety and security should include models such as HAZOP, FMVEA, STRIDE or stochastic modelling (i.e., SAN models and Markov processes). The ISO 26262 standard could also be used as the base of the framework for potential analysis or a guide to developing other models in future based on its requirements. All the models are worth looking into and based on how I wish to analyse (combined or separately) should then choose the appropriate models.

Continuing from this survey of suggested models, (Schmittner, Ma and Smith, 2014) tell us that the existing technique of FMEA only looks at faults and fails to account for security, which is becoming increasingly important, especially in vehicles, so FMVEA, which considers vulnerabilities is proposed as a better choice. This looks at the failures, the vulnerabilities, and their effects. Following the principle of FMVEA, the authors focus on a specific component of the AV in this case, the telematics unit, due to its connection to both buses, allowing potential access to safety-critical systems. In the context of the telematics unit, over-the-air (OTA) updates and Bluetooth are considered to have the highest risk, having high severity and attack probability (their product is the risk), with the possibility of an attacker taking control of the vehicle. This form of risk analysis, or other models such as STRIDE (Schmittner, Ma and Smith, 2014) could be used to prioritize the level of risk to see which attacks to focus on and considering defences for. The focus on telematics also gives us a potential starting point for modelling attacks, especially remote attacks.

Focusing more on the actual AV, the authors of (Shalev-Shwartz, Shammah and Shashua, 2018) tell us that one of the biggest issues with AVs lies in the lack of safety guarantees. The accident rate must be lower than that of a human, which is stated to be 10^{-6} per hour, so the accident rate should be significantly lower than this, ideally around 10^{-9} per hour. RSS is introduced as a system that follows the idea of a duty of care, meaning to exercise care and caution when driving on the road. By choosing appropriate parameters these rules can be implemented in the CAV, in a way that is sound, useful, and verifiable.

RSS rules use consists of the following ‘common sense’ rules for vehicles (Shalev-Shwartz, Shammah and Shashua, 2018):

6. No hitting from behind
7. No reckless overtaking
8. Right of way must be given, not taken.
9. Be extra careful in limited visibility.
10. Avoid accidents wherever possible if it can be done without causing another.

If the model follows these rules, I could even create my own where needed provided it follows these rules well in an effective manner with plausible parameters. The white-box nature of the RSS model means its inner workings are visible (Shalev-Shwartz, Shammah and

Shashua, 2018), which could potentially be exploited with the right kind of attacks which may even manipulate the information being received to check if the rule is being applied correctly.

However, it was argued by (Oboril and Scholl, 2021) that the RSS model in its current state may lead to sub-optimal results in extreme cases and suggest their own, proactive extension of RSS known as RSS+. AVs are simulated using a vehicle simulator known CARLA (Car Learning to Act) when implementing RSS+, as RSS sensors are integrated into the AV in this case, so that RSS and RSS+ could be compared. If comparing potential safety monitors and the vehicle when on the road, CARLA could be a very good simulator for us, as I could load my own maps, libraries and scenarios (i.e. hazards). The simulations show that when comparing RSS and RSS+, the distance is better kept over a period of time for RSS+ than it is for RSS in scenarios such as obstacles, pedestrians crossing and vehicles overtaking (Oboril and Scholl, 2021). Overall, the collision speeds are lower for RSS+ than they are for RSS which indicate better safety monitoring. From this I can conclude that RSS+ is better at proactively dealing with safety on the road, and I could even attempt to model it as a means of mitigation for safety-related issues, though tying it in with security challenges will present its own range of issues.

(Popov *et al.*, 2022) states that AVs need to have fundamental safety issues resolved, particularly the overall safety, going on to model a vehicle experiencing hazards on the road. It is stated that 10^{-5} is the highest the accident rate should be, if considering human drivers, for AVs to be considered safe in comparison. It is impractical to try and derive arguments for safety from mileage due to time and software updates. A better idea is to model the vehicle and its components, using simulation to see the overall safety. The model can be used to see how the failure rate (i.e., risk of accidents) is for a vehicle with or without hazards (i.e., traffic and harsh weather). All hazards are assumed to have a duration, meaning they do not go on forever and the hazards occur at a fixed rate; they are either correctly perceived or overlooked in this case. Parameterization was done through sensitivity analysis and the time of each simulation being 100-9100 hours.

Results from the cases tested are as follows (Popov *et al.*, 2022):

7. Overlooked hazards are shown to have significant effect on the risk of an accident.
8. The duration of such hazards has negligible effect.
9. The risk of accidents following an overlooked hazard that is later correctly perceived are only slightly more, with 3-3.5% risk of an accident.
10. Where the safety monitoring is considered imperfect, effects are significant, with an accident being imminent with 9100 hours, as 40-50% accident probability has been reached by 2000-3000 hours, so the safety monitors must be reliable, with rates of 10^{-6} at most.
11. False alarms (i.e., detecting a hazard where there is none) are also negligible, remaining under 3% for false hazard probabilities between 0.1-1%.
12. Combining imperfect safety and falsely perceived hazards has little different from that of the imperfect safety monitor alone, suggesting that a falsely perceived hazard is of little importance either way.

It can be concluded that while some parameters only have minor effects, those related to perception and planning have the most significant impact, namely overlooking hazards or

imperfect safety monitors, which mean these are where designing and choosing parameters will be most important.

Despite quantifying the level of safety, (Popov *et al.*, 2022) did not address the security component, which is very significant in modern vehicles. A stochastic model for both safety and security was created by (Popov, 2015) using Mobius for an ASIL-D security requirement device known as the e-Motor, with fault tolerant mechanisms, having two channels that detect their own failures, with their own safe and unsafe states. Monte Carlo simulations were used in a range of 1000-12000 hours, with parameters for the repair rates (safe and unsafe states) of the device being obtained through sensitivity analysis, though the analysis itself was not presented. For the simulations, as the repair rates were increased, the failure rates decreased, with unsafe failures happening later and later for a total of 200,000 simulated hours per experiment (Popov, 2015). It is suggested by the authors that probabilistic modelling is the best way to quantify the effects of cyberattacks, with probabilistic modelling being very suitable. The comparison of IDS and individually protecting devices in AVs seems a promising area to explore, which I may do in the future.

I note from (Bloomfield *et al.*, 2021) that assessments of safety for autonomous systems such as AVs, the system should adapt as needed according to what is required of it, with good analysis and testing to give confidence in its safety. The models, stochastic or otherwise should allow us to test the safety in combination with security so that I can determine the confidence in CAV safety and security. Safety modelling in (Bloomfield *et al.*, 2021) was done for a sensor with an ML component and a safety monitor added to it, mostly as templates of the requirements, as well as the strengths, weaknesses and environment. Therefore, when modelling vehicle safety and security, considering ML is especially important, due to being the key intelligent component of the system (i.e., provides the autonomous capabilities). As well as this, the adaptation and changes of the system should be considered, as different changes and future requirements could be of importance due to introduction of vulnerabilities and faults in the future.

A guide to HAZOP analysis, mainly designed for various industrial processes and systems, so that hazards when operating can be seen is offered by (Crawley and Tyler, 2015). System parameters are combined with guidewords and checklists to make sure all relevant questions are asked (i.e., consistency, control of the main parameters). Guidewords are simple phrases that have a generic meaning to describe something in relation to the system design. It must be considered that guidewords are not necessarily an effective way of rating risk, which would have to be assigned to the guidewords or done alongside them somehow for it to be properly quantified. As the CAV is a system with various functions and processes, it may be useful to look at risks and answer all important questions, especially those involving safety using such a method.

The authors of (Schmittner *et al.*, 2015) look at safety and security analysis, comparing FMVEA and CHASSIS when being applied to the automotive domain. Due to how computers and communications are becoming more intertwined, the existence of cyber-physical systems and overall systems dependability becomes more important. Major challenges in cyber-physical system development include (Schmittner *et al.*, 2015):

7. Requirements of engineering scaling up with complexity of the systems, as existing methods do not do so.
8. Categories of dependability can no longer be viewed independently.
9. Mass deployment could present issues, especially as dependencies could lead to misuse and malicious interactions (i.e., attacks).
10. Maintenance will be needed to manage new vulnerabilities, defects as well as components becoming worn out. Some remote maintenance will be needed so the risks will have to be considered for this too.
11. Context is important for such systems so it must be monitored.
12. The infrastructure of the systems should be flexible so that it can adapt as well as being easily modified in future.

Authors present a case study of Over-the-Air updates in a vehicle, which consists of the vehicle, remote user, and communication network. The security is looked at in relation to CAN and FlexRay inside the vehicle, which are found to be insufficient in terms of security. The communications of the vehicle are handled by cellular, Wi-Fi or Bluetooth usually. Authors here have excluded non-critical buses such as MOST or LIN in their analysis. Scenarios looked at include tampering with updates, false updates as well as those designed to interfere with the telematics unit. Telematics attacks are designed to impact availability in this case, either by flooding it with large amounts of firmware to crash the communications of the unit or sending false updates to block legitimate ones from going through (Schmittner *et al.*, 2015).

It was found that CHASSIS is good for the initial stages where elevated levels of abstraction and limited system knowledge is available, but FMVEA becomes suitable once there is information regarding system architecture. CHASSIS requires expert knowledge which can lead to results varying as new groups conduct the analysis while FMVEA produces results that are more reusable and comparable due to lack of extreme variations each time (i.e., due to diverse groups having different expert knowledge). FMVEA can be considered better for risk ratings as it tends to be connected to the system model when describing different scenarios (Schmittner *et al.*, 2015). FMVEA is also considered more formal, which means that CHASSIS is also more flexible even if not as consistent, so dynamic systems where threats change, or things are unclear could be better analysed by CHASSIS. It can be concluded that they both have their own weaknesses, with CHASSIS requiring a lot of judgement from experts, while FMVEA still needs to unify failure and threats. For better FMVEA, more threat catalogues are needed as this is how it identifies modes.

In (Kosari, Popov and Roy, 2022), written by me and my supervisors looked at the safety implications of individual parameters when compromising a vehicle where hazards are occurring at a fixed rate. It was found that individually compromising components is insufficient, so more than one being compromised is the way to go forward in any future modelling or experiments of this sort with Mobius.

A more recent incident was that of Tesla with their full self-driving (FSD) feature going wrong in a tunnel, when the vehicle changed lanes and abruptly stopped as if having identified a hazard such as an obstacle, which then induced a hazard in the form of a rear-end collision which then lead to a pile-up of 8 vehicles which then also resulted in traffic coming to a dead stop on its way to the tunnel for over an hour (Kay, 2023).

Vehicular Communications

This section is more about the communications of the vehicle with the outside world, with the important nodes and communication protocols needed to do this, as an increased amount of information from the surroundings, such as that from vehicles will improve the overall accuracy of the CAV when driving. Vehicles have an on-board unit (OBU) which allows for their communication with other vehicles and more (El-Rewini et al., 2020).

Vehicle-to-Vehicle (V2V)

This is a short-range form of communication between vehicles, exchanging information such as speed and trajectory, which could be used by safety-critical applications, so that vehicles can communicate and coordinate with one another to make more accurate, safer decisions that may even help improve overall traffic (Dibaei et al., 2019; Fowler, 2020).

Vehicle-to-Infrastructure (V2I)

Vehicles should also be able to communicate with infrastructure, which includes information on traffic obtained from roadside units (RSUs). Information obtained from RSUs and exchanged with vehicles helps to understand the upcoming traffic situation, such as congestion as well as changes in different traffic signs or signals (Fowler, 2020).

Vehicle-to-Everything (V2X)

V2X encompasses all communications, including vehicles, infrastructure, pedestrians (Vehicle-to-Pedestrian), remote servers (Vehicle-to-Cloud) and more (Fowler, 2020). Cellular communications are also part of this, being linked to the telematics unit as well as being a specific component of V2X known as C-V2X, meaning it can make use of nearby mobile base stations for communications (Fowler, 2020).

Vulnerabilities

Here, I will look at the vulnerabilities that could be exploited by an adversary to hamper the performance of the vehicle, alter its behaviour, and even take control of it, with the focus being those that could have safety implications.

Attack Surface

The attack surface is how the attacker aims to penetrate the systems of the vehicle and gain access to further resources inside the vehicle. This is what is exposed to the outside, where the attack can either be used to cut off access to the outside or to manipulate the vehicle and its systems. Looking at Figure 5, the attack surface can be described as follows (Malik and Sun, 2020):

6. Remote connections, such as V2X communications, phones, and Bluetooth devices.
7. Ports such as OBD-II and USB.
8. Keyless entry and remotely linked applications (i.e., vehicle apps).
9. Manipulation of ECUs such as steering, transmission, airbags, and lighting.
10. Various sensors or monitors such as those monitoring tyre pressure.

In (Gao et al., 2022), the attack surface and communications are identified as follows:

- **V2X:** Updates, Bluetooth, DSRC, Phones etc.
- **In-Vehicle:** USB, OBD-II, CD, TPMS and any ECU that is accessed once in the in-vehicle network.
- **Sensors:** Camera, LiDAR, GPS, etc.

The GNSS is what provides the GPS data for the localization stage of AVs. The next stage is the perception which uses the sensors, followed by V2X where all the information needed for route planning is reached followed finally by the actuators that act as the actual controls. While any of the elements from in-vehicle can be used to invade CAN, it is also possible to perform a TPMS invasion which involves injecting attack codes into the TPMS which will activate at a certain pressure level, which can act as a sort of sleeper agent.

The proposed security methods for CAN in existence involve either encryption or authentication measures, intrusion detection (i.e. machine learning). In 2012, CAN FD was introduced as a potential future measure by Bosch, as it offers safe data transfer and error detection. However, it does not guarantee confidentiality and integrity, so attacks against these elements in the CAN are still possible in CAN FD.

Examples in the real world of safety and security failures are as follows:

- Sensor security failure in May 2016 causing a Tesla on autopilot to crash into a truck. A similar issue occurred again in June 2020.
- A failure in March 2018 of the operation system security meant that despite detecting a cyclist early on, an unmanned Uber vehicle crashed into the cyclist as the software failure to correctly predict trajectory.
- V2X communications for a Tesla were compromised in November 2016 when researchers successfully hacked a phone, obtaining the login of the user's Tesla app login, giving them remote access to the vehicle.
- In January 2018, the datacentre of GoGet was hacked, stealing a large amount of vehicle user info from the carsharing company.

Attacks

There are various attacks that could affect the functioning of a CAV, with the safety and security issues being primarily due to attacks that compromise the integrity and availability properties of the system. Attacks can be physical, requiring direct access to the vehicle or remote, meaning they can be done across exceptionally long distances. The general form of attacks consists of the following (Dibaei et al., 2019):

10. **DoS/DDoS:** An attacker can flood a node with copious amounts of information to use up its resources so that legitimate nodes cannot communicate with it. In a DoS attack, this typically uses one IP address whereas in DDoS multiple IP addresses can do this at once, being much more effective much more quickly.
11. **Black-Hole:** Packets are dropped so that they do not reach their destination, which means that an attacker can block off parts of network communication, which can be even more dangerous when packets are blocked at a critical point in the network, especially if these communications are intended for safety-critical systems.

12. **Man-in-the-Middle (MitM):** This attack is where a node is between two communicating nodes, without them being aware of this, allowing it to intercept, drop and modify packets at will.
13. **Replay:** A continuation of MitM, intercepted packets can be forwarded to another node at a later point, even being used by an attacker to authenticate themselves as another node, allowing it to be used for impersonation.
14. **Sybil:** This attack involves creating multiple identities that can then allow the attacker to have more control of network resources as well as greater influence which could allow for manipulation of other nodes in the network by sending them information to change their behaviour (i.e., with multiple false CAVs causing another vehicle to change its direction because of the information).
15. **Impersonation:** Quite straightforward, in which the adversary pretends to be another, legitimate node and is now concealed in the network, having the same level of access and authenticity that the legitimate node enjoys.
16. **Malware:** A malicious node in the network could install software containing a range of malicious tools such as Trojans or viruses into the systems, which can alter the functionality of the system, and even propagate from the vehicle to other nodes using the communications.
17. **Falsified Information:** Information that is not correct is spread through the network, usually through communication systems, and could be propagated throughout a network if not detected, affecting the decisions of nodes involved, which can be very serious if it affects the driving decisions of CAVs on the road in real time.
18. **Timing:** An adversary using a rogue node can inject messages with time slots to increase delay, which can have serious implications for time-sensitive applications as they will now receive information later, resulting in later reaction times which could increase the risk of accidents.

Attacks are not always direct, as the adversary usually goes through the attack surface, and depends on what they are aiming to compromise. In order to do so they could perform a range of attacks that involve compromising different areas, but will either disrupt sensors, machine learning or aim to go for the in-vehicle network, which can be done through the OBD-II port, USB, communications or the telematics unit, a specific ECU connected to the entertainment and the CAN (Foster et al., 2015; Pham and Xiong, 2020). Looking at the vulnerabilities in Table 4, there is clearly a need for more work as well as the implementation of defences, especially as there are still some knowledge gaps when it comes to security, including the following (Parkinson et al., 2017; El-Rewini et al., 2020):

5. Vulnerabilities in telematics systems.
6. Implications of exploiting navigation systems (i.e., compromising GPS).
7. Analysis of the vulnerability and effects of compromising individual ECUs.
8. Defences for LIN, FlexRay and MOST, as well as defences to prevent attacks from different buses reaching each other (i.e., LIN vulnerability being used to get to FlexRay).

A specific case of attack through a unit attached to the in-vehicle network covered by (Foster et al., 2015) was the telematics unit, which is considered an aftermarket device (i.e. added to the vehicle later by a separate manufacturer). The authors examine the telematics unit,

particularly a unit connected to the OBD-II port, showing that it can be discovered and attacked remotely. Like most ECUs, the telematics control unit (TCU) can communicate with the CAN bus meaning if compromised it can access the in-vehicle network. If physically accessing the TCU, it is possible to take it apart, even download information it contains to access the vehicle systems in future using any credentials or passwords stored on it. Remote attacks focus on the cellular network and IP, as both can be used to send and receive requests from the TCU. These included internet-based attacks in which previously identified details could be used to log into the unit, or to use SMS to initiate rogue updates or installations that could be used to install a backdoor on the unit to access it at any time, even erasing all traces of the update afterwards. Devices could be found provided that the IP or phone number of the TCU is known; this can be done using search engines to find the ranges of IP addresses for possible TCUs, as well as the fact that certain area codes for phone numbers are reserved for TCUs in some cases which can allow for random testing and dialling to find them. By compromising the system, it is then possible to send rogue CAN packets into the in-vehicle CAN bus, which the author found that functions such as brakes and windscreen wipers could be controlled. Solutions to the TCU vulnerability included authentication of updates, better SMS security (i.e., use of blacklisting or whitelisting to control which numbers are allowed to communicate with the TCU), and even disabling administration via the network which can prevent any form of internet-based attack.

Defences

This section looks at some existing defences that are either being fully implemented or are in progress, as well as their benefits and drawbacks. The limitations will be looked at as well as where they could be implemented in CAVs. These include cryptographic solutions, which consists of encryption and authentication, as well as methods to detect and prevent attacks.

Authentication

With lack of authentication comes the ability to impersonate nodes and send messages without any way of checking where they come from or their legitimacy in general. This is a particular issue in the in-vehicle network with the ECUs and the bus overall.

This also means that firmware updates could potentially be loaded into different subsystems without any approval needed, so cryptographic methods should be implemented in which ECUs or other nodes must authenticate themselves with a central authority that manages security using asymmetric encryption, after which communications are then set up via stream authorization which uses symmetric encryption for messages (El-Rewini et al., 2020).

Different authentication protocols have been presented in the past, including the use of hardware-based authentication as well as Message Authentication Code (MAC) being added to frames in the CAN bus in order to authenticate messages sent by ECUs in the CAN in real-time (Pham and Xiong, 2020). By using methods to securely verify the credentials of any sender on the network, attacks such as impersonation and false identities can be prevented (Chowdhury, Islam and Khan, 2019).

(Shangguan *et al.*, 2022) proposes dynamic watermarking, where controllers, such as those in ECUs add random private signals that are treated as a sort of background noise. Sensors can report these measurements and any changes in these background values could be used to detect a potential attack such as with false messages where the watermarking has not been

done. The experiment done by the authors aimed to target a specific node, tampering with the original data input, and sending a false measurement to a controller. This attack could be performed against various components such as Bluetooth, Wi-Fi, and telematics. Tampering with data, in this case the yaw rate for different periods of time showed that the watermark-based detection method was very effective, being detected within 20ms when the sample rate for the detector is set to 50Hz; it is implied that this can be made faster for higher rates.

Intrusion Detection Systems (IDS)

A means of securing the overall networks for communications, both for in-vehicle and the wider network of vehicles on the road (i.e., VANETs) is the IDS. The IDS is a system based on deep learning which monitors the network, being a trained ML model, typically a recurrent neural network as this can compare to check for unusual behaviour that could be the result of an intruder trying to access the network (Chowdhury, Islam and Khan, 2019; Dibaei et al., 2019). There are two main types of detection used in IDS (Dibaei et al., 2019):

- **Signature-Based:** The IDS is trained with known attack scenarios and their signatures or patterns, which mean that the behaviour of the network can be compared against known signatures of adversaries to detect them. This is mostly for known attacks and will not be effective against new attacks that the model was not trained with.
- **Anomaly-Based:** Instead of looking at known attacks, this type of IDS will determine a baseline for normal behaviour in the network, such as the traffic, messages sent and the time intervals between communications, so that behaviour that deviates too much from this baseline can be deemed abnormal, being classified as a potential attack.

Detection systems are found to be effective, though depending on the algorithms implemented, the amount of overhead can be extremely high, as well as the fact that it is possible to fool ML models, as discovered from adversarial examples. In addition to intrusion detection, it is also important to detect software vulnerabilities and the presence of malware, for which similar models can be employed (Dibaei et al., 2019).

(Sharma and Möller, 2018) look at the importance of protecting the in-vehicle network, especially the ECUs which govern different vehicle tasks, proposing an approach for intrusion detection and its subsequent prevention, known as IDPS. The in-vehicle bus is assumed by the authors to be the CAN bus. The IDS on its own simply monitors and detects potential attacks based on strange behaviour, whereas the IPS is the ability to respond to a potential intrusion, which then combines to form the IDPS. IDS either monitors internal activity, network activity, or even a mixture to detect unusual activity that could indicate an intruder. Detection is done either via signature (i.e., patterns) or behaviour, meaning unusual patterns or behaviours that may not be known, which can be useful especially for unknown or unpredictable attacks. IDPS uses neural networks and the relevant datasets to detect various attacks such as DoS and impersonation. IDPS in these cases used k-NN (Nearest Neighbour) classifiers, giving the following results:

3. For CAN traffic logs used as training datasets, 80% classification accuracy if the information was from a dataset, 68% if it was unknown (i.e., from outside the training data).

4. Use of SVM resulted in 82% accuracy for dataset information, and 59% accuracy when outside the datasets.

The authors proposed that recurrent neural networks (RNN) be used for the IDPS architecture, with past data being used to train the model so that it can predict future data packets, which works well especially due to limited vehicle data knowledge in the context of cybersecurity. The neural network predicts a packet and then compares it to the actual packet, constantly feeding this difference back into the system so it may learn, with big differences being considered anomalous. The IDPS can be considered a work in progress and should be able to detect in real time, with low error rates and low power consumption.

Intrusion Tolerance

Systems are considered intrusion tolerant if they remain available and retain security properties even if some of the components are compromised in an attack, typically using a protocol called Byzantine Fault Tolerance (BFT), which states that provided all of the components or replicas are not failed, the system will keep working, but in intrusion tolerance these failures/faults are assumed to be the work of an adversary (Bessani, 2011). The intrusion tolerance algorithm in its most basic form can be described in this way:

5. Let there be a value m which is a positive integer greater than or equal to 1.
6. For this value, the number of participants as per the Byzantine agreement will be $3m+1$ (e.g. for $m=1$ there will be 4 participants).
7. The functional property and system security will be maintained so long as an adversary does not overcome $m+1$ participants or more.
8. The participants, also known as replicas, will be cleansed periodically to move them out of their failed states through some predefined cleansing rate.

In order for intrusion tolerance to work properly, the following properties are important (Bessani, 2011):

5. The system must work well under all conditions, meaning it should be robust, so that even when under attack it keeps working well.
6. There must be diversity in security measures or otherwise, especially as common vulnerabilities are a weakness in the protocol.
7. The system must detect and cleanse intrusions to make sure that the adversary does not continue compromising replicas throughout the system until they eventually compromise them all. The attacks must be detected and stopped where possible.
8. Confidentiality, while not vital, could be a requirement in some cases.

By periodically rejuvenating multiple replicas of the same system or its software, it is possible to always keep parts of it available. If for a number of replicas, f out of $3f+1$ are compromised, then the system will still remain available, giving time to cleanse as well as remove any traces of an adversary (Platania et al., 2014). Replicas can be assigned the following states (Platania et al., 2014):

- **Correct:** The replica is working completely fine.
- **Compromised:** The replica is not behaving according to the algorithm, showing arbitrary behaviour.
- **Crashed:** The replica has stopped working and needs to be restarted.

- **Rejuvenating:** The replica is being reset, so compromised or not it will become a crashed replica and be reloaded.
- **Partitioned:** A replica that is considered separated from other replicas due to its inability to communicate with enough replicas.

Diversification of replicas allows for them to be defended across space, as they cannot all be vulnerable in the same way as their attack surface may alter, so there is no guarantee of compromising one and therefore being able to compromise another. However, with enough time, an adversary could still learn to compromise the diverse replicas, so the replicas should be periodically rejuvenated and diversified in such a way that they are not similar to any other replicas in the past or present, which is defence across time (Platania et al., 2014). For a system where it is hard to both know and mitigate all attacks while also maintaining availability, such as in CAVs, this could prove to be a good defence to model for cyber-resilience.

(Bessani, 2011) looks at being intrusion tolerant, compared to fault tolerance. While fault tolerance maintains availability, intrusion tolerance means security properties are maintained as well. Replication is already found to be effective both in terms of performance and minimal use of resources, but more work needs to be done on the effectiveness of diversifying replicas and making protocols more robust. The author makes it clear that open problems include the lack of definitive rules on how the system should react to intrusion, as well as making sure that state transfers in intrusion tolerance are time-bounded in a way that makes sure the adversary cannot delay the protocol to buy themselves more time to compromise a greater number of replicas.

(Sousa *et al.*, 2010) note that proactive recovery has been used to improve the resilience of intrusion tolerant systems in the past, making sure that the minimum replicas are present to keep the system available and working properly. The authors of the paper look at how proactive recovery could improve intrusion tolerance now. If faulty behaviour of a replica is detectable, the replica will be recovered as soon as possible (which is important as the fault should be fixed as soon as possible), and the implementation of intrusion tolerance with proactive recovery was indeed able to increase resilience, being effective at dealing with DoS attacks both from inside and outside the network.

Cyberattacks are increasing, particularly against critical infrastructure. Therefore, (Platania *et al.*, 2014) present an intrusion tolerance system which defends across space and time, with diversification as well as proactive recovery. It uses the BFT protocol as is the case with all intrusion tolerance, and showed 95% confidence over a 30 year period for the intrusion tolerance model even for a 1TB state transfer (Platania *et al.*, 2014). The proactive recovery works for large states, with fast data retrieval and low bandwidth usage as the two strategies. Proactive recovery and diversification protect against attacks but not against flaws in the protocol itself such as those that make it vulnerable to more attacks. Replicas created of the software in parts of the system are diverse (i.e., similar but different, so no one attack works on all), and these replicas are either correctly working, compromised, crashed, or rejuvenating (i.e., being reset/cleansed). A replica could also be partitioned so they are separate from the others until they are working properly, as they cannot communicate with enough replicas in this state. The adversary was assumed to be immensely powerful with the ability to compromise replicas but not affect communications if they are working correctly,

with no physical access to the system. It cannot bypass any implemented cryptographic methods either (Platania *et al.*, 2014).

(Popov, 2017) looks at modelling cyberattacks in software, with the idea that failure regions (i.e., states where software no longer works properly) are introduced when the software is compromised. The model architecture looks at fault tolerance, with different software that could be used to protect systems, with replicas being used to implement Byzantine fault tolerance (BFT) protocol. Methods such as patching and cleansing are to be used for the maintenance model of the software, with different attacks such as independent or synchronized attacks, and showing that reliability is affected by the different attack models. The authors model a 2-channel software system with different maintenance policies and uses a SAN model to create it as a network for simulation. For the 2-channels, 1-out-of-2 is used meaning that as long as one of the two channels is not compromised/failed then the system is available and can continue operation. The attacks in the model of the author are not ones that cause instant failure, but successful attacks that reduce reliability (i.e., advanced persistent threats). Cleansing is found to have one basic issue, which is that if a threat has previously compromised the software, even if it is removed, it is possible to do it again, even in a shorter amount of time. Patching can be costly, as well as potentially causing further issues if the patch is not done properly, so the better of the two should be chosen. Therefore, the author has chosen to compare cleansing and patching to see their effectiveness relative to one another. The experiments in question, created in Mobius 2.4 are as follows (Popov, 2017):

5. 2-channel system without maintenance (no cleansing or patching).
6. 2-channel system with cleansing only.
7. 2-channel system with patching only.
8. 2-channel system with cleansing and patching.

The simulations were set to be around 350 days at 10-day intervals, and calculation of the failure probability was done for 50-day intervals, repeating studies three hundred times. If the required confidence level of 95% in the results was not reached, then repeats were done five hundred times instead. Cleansing and patching are first compared in the case of independent attacks, with maintenance rates of 0.01 (around 15 minutes). In the absence of maintenance, failure rate was shown to grow. For the first 250 days cleansing does better than patching, after which patching is better at reducing failure rate (Popov, 2017). When combined, as expected both forms of maintenance together rapidly reduce failure rate. Next, the author looks at the effects of synchronized vs. independent attacks, with independent attacks being more successful but less likely to cause failure than synchronized attacks. Both patching and cleansing alone seem to be less successful for synchronized attacks, with combined maintenance methods causing the system to behave as it would under independent attacks, which is preferable as this means at least one channel is kept available, meaning that patching and cleansing complement one another and should ideally be modelled together. By using deferred patching (i.e., making sure there is a problem before patching), solutions can then be implemented when needed rather than constantly applying patching. The results show that cleansing could be very effective in countering attacks, and patching depends on the vendors, so it cannot be adapted to system needs as the system designers cannot control this (Popov, 2017). Results suggest that deferred patching in fault tolerance could be better, but this requires more detailed exploration. Therefore, the system could work with a focus on

cleansing while also having deferred patching. The author focused on fault tolerance but proposed extending the work to intrusion tolerance to assess its effectiveness against different attacks.

An architecture for intrusion-tolerance in Avs is proposed by (Vöelp and Esteves-Verissimo, 2018). The vehicle has diverse sensors and replicas, with sensor fusions and the pre-processing of visual data. This then goes into a capsule network which has both the complex manoeuvres, and the fail-safe linked into a safety kernel (for deciding how to perform a move) which allows for a rejuvenation process of all replicated components of the AV. The two vulnerable areas indicated here were the AI stack and the vision, suggesting that the perception and ML components of the vehicle are where it is best to apply intrusion tolerance. (Singh, Cukier and Sanders, 2003)

(Singh, Cukier and Sanders, 2003) state that intrusion tolerance is an emerging approach to allow systems to continue working when compromised by introducing replication. This paper attempted to validate such a security measure by creating SAN models to check the efficacy of such an idea before it can be deployed. A major source of uncertainty in security validation will always be the behaviour of the attacker as this is often unpredictable and an unknown that is hard to model depending on the type of attack. It is said that all models of intrusion tolerance need to include sub-models of the attacker, the intrusion tolerance mechanism as well as the system and its states. The SAN model used included one for a replica, a host, and the manager that controls the replicas (i.e. cleansing or otherwise). These are all joined together in a composed model which can allow different parts to be increased or decreased via the 'Rep' function. For each domain it was found that putting few hosts was a good idea, as intrusion tolerance in this case was very sensitive to how many domains were available to start new replicas.

New attacks are constantly being developed against systems with the aim of corrupting components and violating the integrity property of security as discussed in (Pal *et al.*, 2006). This has not been addressed much at the time of this paper, so the ITUA architecture has been proposed which involves integrating multiple defence mechanisms into a security architecture with auto-adaptive intrusion tolerance to ensure as much security as possible. Intrusion tolerance itself aims to maintain operational capability when the system is under attack, especially as many of these attacks tend to result in corruption or loss of control of various resources (e.g. CPU cycles, memory, bandwidth, processes, and hosts).

The authors in (Pal *et al.*, 2006) created a prototype model to test for adaptive intrusion tolerance for the ITUA architecture to see its tolerance to attacks, with managers for the security and replicas each being contained within a relevant security domain which creates boundaries making it harder for an attacker to have direct access to each part of the system just by getting access to one area. Replicas are also started in a probabilistic way by managers to make sure a compromised manager in the system cannot take control of how replicas are placed or started, and all replicas have public keys which are known by all those who interact with it to ensure it is legitimate.

The ITUA architecture contains a rapid reaction loop where a quick response to attacks is made based on information from a sensor and IDS, with the loop monitoring the effectiveness of the response to the attack as well as the type of attack (e.g. same attack or multiple attacks, etc.). It is also possible to isolate hosts and domains by suspending trust based on observed

behaviour, removing them from the architecture. It was found that auto-adaptive intrusion tolerance is possible, as the system can now adapt to an adversary and create various responses that the attacker must overcome, especially given that there were good response times by all parts of the system. This was said by the authors to be the best method of surviving attacks as it combines protection, detection and adaptation (Pal *et al.*, 2006).

The main contributions by (Pal *et al.*, 2006) involved an auto-adaptive intrusion tolerant architecture, and the information gained about it based off experience and testing (i.e. it was very effective). Techniques and mechanisms included the adaptive form of BFT replication, where corrupt replicas are removed and replaced with new ones, graceful degradation, host redundancy (i.e. extra hosts where needed) with a decentralized architecture, injecting uncertainty into responses and the integration of multiple defence mechanisms into a single architecture.

Security

This section of the literature looked at the security components, namely attacks and defences against CAVs, consisting of overview/survey papers as well as any analysis of such attacks that was in existence.

Overviews

The authors of (Scalas and Giacinto, 2019) look at the foundations needed for CAVs. The necessity for a security-by-design approach (i.e., considering security during the design process rather than treating it as an afterthought) is stated as well as proposing that knowledge of all core cybersecurity aspects must be known, such as in-vehicle vulnerabilities and potential solutions. Constraints include limitations of hardware such as limited memory and low processing power, as well as the time-sensitive nature of CAVs due to being a real-time system. Access to the in-vehicle bus, especially in the case of CAN is considered a major threat due to its history of having no implemented security measures, which can allow manipulation of ECUs connected to it to control the vehicle. It is assumed that goals of an attacker include but are not limited to stealing either the vehicle or its data, or to install some sort of malware to alter the behaviour of the vehicle. Some proposed solutions by the authors include special hardware to deal with security such as encryption, authentication, and intrusion detection. Eighty-four percent of professionals working with OEMs at the time of this paper were said to have serious cybersecurity concerns. It is stated that the concerns such as those with the in-vehicle network could be further analysed to better understand driver and communication security, which I could aim to do by expanding my models of CAV cyber-resilience (Scalas and Giacinto, 2019).

The survey by (Pham and Xiong, 2020) briefly outlines components such as GPS, LiDAR and the protocols for different systems in CAVs. Attack targets were outlined as the OBD port, ECUs, CAN bus, sensors (i.e., LiDAR, radar, and cameras), the GPS as well as the communication systems (i.e., V2V and V2I). Attacks covered are either remote or physical, with the aims of gaining control of the vehicle, interrupting vehicle functions or to steal data. All attack models aim to somehow reach the CAN bus, with all of them being for remote attacks, but OBD being aimed at physical attacks (i.e., hands on access to the vehicle). The main defences included intrusion detection (IDS) and the use of encryption, which are worth testing in my models for cyber-resilience. Research challenges indicated include unsolved

attack models (i.e., no defence for the attack), as well as the lack of realistic experimental data for different attacks and defences.

Here the security challenges of vehicle communications are presented, with in-vehicle components such as CAN, LIN, FlexRay, MOST and Ethernet. Specific areas such as the telematics unit, ports as well as the EV charging infrastructure are also addressed to consider the electrification of vehicles that is gradually occurring. Communications for vehicles mentioned by the author are V2X, DSRC, C-V2X, Bluetooth and more. AV framework is described into the following 3 stages (El-Rewini *et al.*, 2020):

1. **Sensing:** This consists of the various parts of perception, mostly sensors such as LiDAR, camera, and GPS.
2. **Communication:** The in-vehicle buses such as CAN and FlexRay, as well as communications and the relevant exploits for systems that can connect to the in-vehicle network.
3. **Control:** This is the critical part of the vehicle, consisting of brakes, motion planning, and steering.

The authors show that as part of common in-vehicle architecture, the engine systems are connected through the CAN bus as part of the powertrain, while safety is connected via FlexRay, meaning that the CAN bus and FlexRay are areas no attack should reach to keep the vehicle safe (El-Rewini *et al.*, 2020). Therefore, modelling the attacks and defences in these parts of the vehicle is quite significant. Attacks include but are not limited to impersonation, DoS, and injecting data. Some relevant research gaps I find include defences for FlexRay, vulnerabilities in telematics and defences against DoS attacks.

(Chowdhury, Islam and Khan, 2019) continues the theme of vehicular communication by noting that the development of vehicles, particularly their increased connectivity to infrastructure and other vehicles results in a cyber-physical system where new attacks are introduced. The authors here explored the attack surface, vulnerabilities as well as potential methods of detection and prevention. Security issues could lead to an attack that allows access to the vehicle control system and even cause fatal crashes, making security a vital point. The attack surface is divided into the in-vehicle systems (i.e., sensors, software, and other in-vehicle components), the communications (V2V, V2I, V2X) and supporting infrastructure. Some of the biggest concerns in CAV include over-the-air updates and bugs in software installed in the ECUs of the vehicle (Chowdhury, Islam and Khan, 2019). Good security should consist of cryptographic solutions, such as secure updates. Suggested solutions for trusting a vehicle (i.e., avoiding being affected too severely by a compromised node) include a reputation-based system in which priority is given to those nodes that consistently exchange correct information that causes very few problems. The future landscape will consist of AI, specifically in machine learning, software-defined networks (SDN), and network function virtualization (NFV). The use of SDN and NFV means that there is less of a hardware dependent network, and that the mitigation methods can be deployed straight into the network via the software component, as the network and its controls can be deployed through SDN (Chowdhury, Islam and Khan, 2019). The authors conclude that detection and mitigation are areas of key research, with many challenges to still address. With security concerns including acquisition of information, V2X is pointed out as a

particular concern by the authors as an area that needs more security, as it can also allow for the propagation of attacks.

V2V security has had some various proposed solutions, especially to attacks such as DoS, false messages and black-hole attacks. The solution in (Alawieh *et al.*, 2022) is for each vehicle to monitor its neighbouring vehicles, so that rogue or false nodes can be removed from the routing table of a vehicular network to remove their malicious influence. The protocol used was AODV, for 100s simulation duration for traffic between 20-100 vehicles in size, using the 5.9GHz frequency band that is used for DSRC in vehicles. When attacked it was found that vehicles experienced a massive increase in delay, going from 70ms to 940ms as more vehicles became affected. When then countermeasure of AODV is applied to remove the rogue vehicle being used to perform V2V attacks, the delay still increases but remains lower, starting at 30ms and going up to 250ms as the number of vehicles increases. The authors suggested that future research in this should look at the effect of different network sizes and vehicle mobility for the same solution.

In (Sedar *et al.*, 2023) there is review of V2X advancements, specifically the security mechanisms and future research that may be needed. A lot of existing literature involves cryptography, attack types as well as authentication. Very few of the literature articles are found to look at machine learning, defence mechanisms, or areas such as network slicing and blockchain in relation to CAVs. The methodology of this paper involved classifying attacks or threats, followed by existing mechanisms followed by investigation data driven techniques due to the presence of AI in CAVs. Trust and verification of messages were found to be issues, as well as delays caused by security solutions with large overhead.

With AVs being expected to replace drivers in the future, there are safety and security challenges that need to be addressed, with (Cui *et al.*, 2019) presenting safety issues, attacks and the relevant countermeasures. Potential failures include failures in vehicle components or infrastructure. Vehicle components include software, mechanics, communications, and hardware. The hardware failures can then be broken down into the sensors, actuators, and controllers, as well as the integration platform (i.e., integrating components properly). Infrastructure failures can be due to road users, weather, construction, road conditions, and traffic signals/signs, all of which could cause hazards or fail. Safety measures can be based on segment (i.e., on parts of the road, such as with a pedestrian barrier or speed limit), or intersection based (i.e., phases for pedestrian crossing and vehicle turns so everything is better organized). Increased visibility, signals, and the introduction of bays for vehicles that need to turn left later can also work as good countermeasures. Barriers and increased visibility reduced incidents by 71%, while bays and phases reduced incidents (i.e., injuries and crashes) by at least 30%. By timing the phases (i.e. splitting the waiting times between vehicles and pedestrians properly) it is even possible to reduce crashes involving pedestrians and vehicles by 60% (Cui *et al.*, 2019). The actual effectiveness of countermeasures still requires more research, and at this point does not cover the role of security or attacks. Attacks were found to be possible wherever there was an exchange of data with the outside world, such as with infrastructure, sensors, or external devices (i.e., mobiles). Mitigation methods included anti-spoofing, anti-jamming, intrusion detection, data filtering and improving the overall security of the in-vehicle network. Another method included separating the infotainment system from other components of the vehicle, suggesting that the current setup in vehicles makes the infotainment system a serious vulnerability point. Challenges for

vehicle safety include the lack of proper in-vehicle security, as well as communications (i.e. for smart cities and vehicles), particularly V2X are worth looking into (Cui *et al.*, 2019).

The authors here do an overview of both attacks and defences in CAVs, with focus on the major attacks and their potential defences. Examples of successful attacks on vehicles include a Jeep hijacking in 2015 as well as the hijacking of a Tesla Model S and BMW. The authors described three types of in-vehicle architecture (Dibaei *et al.*, 2019):

1. **Gateway-Based:** Vehicle where all ECUs are connected through a single gateway, meaning they must all share the same bandwidth.
2. **Domain-Based:** ECUs are divided into domains according to similar functionality and can have necessary bandwidth allocated to their domain according to need.
3. **Centralized:** All ECUs are divided into zones, like domains, with one central computer to process all information and tasks, which can grant full sensor fusion but also be a single point of failure.

Security requirements outlined by the authors include authentication (i.e., verifying identity of users), integrity (data validation) and availability (ability to respond in real time to as many users as possible). The authors aligned the attacks with the CAV architecture, starting with the E/E, where attacks aim to compromise the in-vehicle bus through a range of attacks including but not limited to DoS, replay and message injection. The communication network elements such as Wi-Fi or cellular were also found to be vulnerable to attacks like DoS and malware. Sensors, especially newer ones can be manipulated with false information via spoofing attacks. The existing defences include cryptography, network security methods such as intrusion detection, malware detection and vulnerability detection in software (i.e. testing the software while it is working) (Dibaei *et al.*, 2019). More code means a larger software with more potential for vulnerability, with some vehicle manufacturers such as Ford having up to 150 million lines of code. Challenges in defences include the massive overheads when trying to implement cryptography as well as difficulties in implementing intrusion detection, especially against unknown attacks, where abnormal behaviour is the key method of detecting an attack, which requires well-defined rules to work properly. The same can be said for malware detection, with more intelligent methods being required in the future (Dibaei *et al.*, 2019). Therefore, looking at the role of detection systems in existing defences, it is worth attempting to model them when testing cyber-resilience to try and quantify their effectiveness. Future directions when looking at security defences should involve lightweight solutions that are software based, with the possibility of using deep learning, which is said to be more effective than regular machine learning methods.

(Parkinson *et al.*, 2017) looks at future challenges in CAVs, with many of the studies done on vulnerabilities tend to be reactive (i.e., after the system is built), with some gaps identified in the knowledge on CAV security. The authors state that CAV is still considered new and has yet to really be subject to adversarial attacks, with academic research still being the base of CAV security, with attacks becoming more popular as CAV technology increases on the roads due to increased financial motivation. Some gaps in knowledge are as follows (Parkinson *et al.*, 2017):

1. Implications of attacks on navigation, which could have safety concerns and grant an attacker vehicle control.

2. Potential to exploit sensors, even changing vehicle function and performance.
3. Required number of sensors for sufficient redundancy, as too few creates blind spots, and too many is costly.
4. How ECUs are compromised and the effects of doing so.
5. Software updates to CAV components such as ECUs, as this could result in rogue updates.
6. Detection and prevention of DoS attacks by CAVs.
7. Implications of large-scale attacks against CAVs, especially automated ones that require little expertise.

These gaps and more indicate that it is possible to compromise the vehicle control, as well as interfering with its functionality and performance. Most knowledge gaps, which also include how the systems (i.e., vehicles and infrastructure) as well as the driver will react in the presence of an attack are all areas which will require extensive research and should be addressed before the number of autonomous vehicles on the market increases.

The focus in (Lu, Qu and Liu, 2019) shifts to network security in vehicular ad-hoc networks (VANETs), where the authors look at the newer advances in models such as those for defences and simulations. Attacks on availability of VANETs include DoS, jamming, malware, false messages (i.e., broadcasting false alerts) and spamming. Integrity attacks include message alteration/injection/suppression, impersonation or replay attacks. I already see that the number of attacks affecting availability is far greater than those affecting integrity, meaning there is a larger range of availability attacks to model compared to integrity, which could make them more common. Simulators used for VANETs to test their effectiveness include mobility and network simulators, with the two simulators linked to see effects of the network simulator on the mobility simulator. This could be useful if looking at safety implications as modelling an attack on the network could then immediately show what effects are present in the mobility of vehicles in the VANET. The authors of the paper believe that privacy and trust management are the key elements of VANET security, especially in securing V2V and V2I which are said to be quite popular.

The VANET is vulnerable to DoS attacks which take advantage of the exposed nature of wireless networking (Hasbullah, Soomro and Manan, 2010). The authors here chose to focus on the effects of DoS on network availability as well as potential solutions. Different attacks could be performed across various parts of the VANET, such as the resources and communications (i.e., V2V, V2I or V2X). By flooding with data, signals or other communications, the nodes such as roadside units (RSUs) or the onboard unit (OBU) of a vehicle could be rendered unavailable. Proposed solutions include changing technologies (i.e., from Wi-Fi to cellular if the network is being DoSed) or frequencies according to availability so that the attack no longer has the desired effect on the system. Threats such as ID disclosure were considered low threats, while false data and Sybil attacks were medium threats. The highest levels of threat were DoS and DDoS attacks, where DDoS is obviously a bigger threat due to more nodes being used to perform the attack at once, using up more resources. Implications of DoS and DDoS are that legitimate users cannot use the part of the network under attack, which will eventually render it unavailable, even causing the network to break down. If critical information is exchanged across the VANET, this could become a safety issue, so modelling it could help us better determine how safe it is to use the VANET

and its communication methods for safety-related communications (Hasbullah, Soomro and Manan, 2010).

In (Greenberg, 2015) the infamous Jeep Cherokee hack of 2015 is discussed, where the hackers conducted a test where they took control of a vehicle with the driver in it. They were able to take control of the following:

- Climate Control
- Radio
- Windshield Wipers
- In-Vehicle Display
- Steering
- Brakes
- Transmission

This test allowed the hackers to commandeer the entire vehicle, even stopping it in the middle of a highway with traffic present, which could be a profoundly serious hazard.

The BeARCAT project by (Fowler, 2020) was for the security of CAVs, designed to work as an assessment and feasibility study for the development of CAV testbeds in 2020. In this work the authors from different companies identified vulnerabilities as well as how to best maintain cyber-resilience. Communications and cloud services are identified as a key area for cybersecurity investment, with the need for a good security framework. At the time of this report the authors also stated that C-V2X could become a global standard for connectivity, but more research is required to see how it would interact with short range communications (DSRC) and that thought needs to be given to what communication system should be used for V2V, with the authors clearly stating that C-V2X and DSRC are unlikely to be used in the near future (Fowler, 2020). Threat models were introduced, which included STRIDE, DREAD, TARA, and PASTA. The STRIDE model was then applied to various vehicular components such as the ECUs, for which elevation of privilege (i.e., unauthorized access and control) was a frequently violated property, making attacks on components such as telematics, radio, infotainment, ECUs, and mobile apps sound methods of taking control of CAVs. Vehicle certification for security is recommended to make all vehicles meet a basic security standard, but there is no way to guarantee it even if this is somehow put into place, as well as the added cost of trying to do so (Fowler, 2020).

This work by (Wei, Razak and Kamis, 2022) looks at the potential attacks against applications in vehicles, mostly availability attacks. Applications can either be those that aim to provide comfort (i.e., non-safety) or those that are critical to improving safety, such as those used in collision avoidance or for warnings regarding lane changes. These require safety messages from other vehicles which aim to improve the driver experience by increasing their knowledge of surroundings, such as vehicle behaviour in V2V. Attacks on the safety systems include DoS, tampering, jamming and malware. Critical safety information may not be received in time to deal with potential hazards and could lead to collisions or accidents. Of the different threats, compared to tampering or malware, DDoS is a considerable risk, and DoS is medium risk, with proposed solutions involving detection algorithms implemented across the vehicle network using machine learning. The authors of

the paper have proposed a method in which packets are processed and checked to detect DDoS followed by appropriate mitigation.

Stuxnet is a powerful piece of malware that can propagate across networks, continuing to devastate multiple nodes. Due to the networked nature of CAVs, it was found in (Ahn, Choi and Kim, 2023) that vehicles can also be compromised by a virus that has an epidemic style of spreading like Stuxnet. This paper mostly looked at the propagation through physical contact, but other forms of propagation are possible, and the effects on propagation for different vehicle brands and types of malwares were not considered here. The mathematical model for the virus propagation was designed to behave like the spread of the malaria virus from host to host.

(Sumra, Ab Manan and Hasbullah, 2011) show potential timing attacks, which add delay to messages to disrupt synchronization or timely communications. Examples included those on V2V, which result in messages such as those warning of accidents arriving late, so that decisions are made later. This can result in the vehicle going towards the accident zone as it is not made aware of the incident on time due to the timing attack, which can be performed on one or more vehicles at the same time. This type of attack can also be used to delay V2I authentication so that a vehicle does not receive services such as warnings from local infrastructure about an accident further ahead, as the authentication messages are delayed so that by the time the infrastructure receives it the vehicle has already gone to the next infrastructure node (i.e., RSU).

The paper by (Yadav, Ansar and Chaurasia, 2022) is a survey of attacks and potential solutions. Elements such as OBD, telematics unit and TPMS are very vulnerable components, as integrity attacks and access to the CAN bus can be achieved through these to adversely impact vehicle safety. The CAV can be divided into three layers, specifically a sensing, control, and communication layer. The attacks can be performed against sensors, CAN bus, replay, DoS, and other integrity attacks involving false information. Blockchain is at one point proposed as a decentralized security system which can allow for secure systems where things can be easily verified.

In the paper by (Shirvani, Baseri and Ghorbani, 2024) focuses more on electric vehicles that are connected, but share similar elements to CAV. The security areas of concern for these consist of the charging station, privacy, software security, connected vehicles and the autonomous driver. The main attacks of concern from an integrity and availability standpoint across all of these tend to consist of spoofing, tampering, DoS, repudiation (i.e. impersonation and eavesdropping) and elevation of privileges.

A more detail review paper by (Sharma and Gillanders, 2022) gives a review of the state of the art in CAV security, breaking CAV into three main sections consisting of sensors, communications and actuators. The vehicle infrastructure consists of in-vehicle, followed by the communications (e.g. V2V, V2X), which is then followed more advanced features such as collision avoidance, speed enforcement, and parking. Up to the communications part, this is merely the connected vehicle, but once you manage other vehicles on the road and perform tasks, it becomes the intelligent transport, which is a much wider area to cover. The sensors consist of the following:

- Camera

- LiDAR
- RADAR
- Ultrasonic
- Wheel encoder
- IMU
- GNSS

The actuators are the physical controls of the vehicle controlled by CAV, such as the throttle, steering and braking. Security is deployed across all layers which use the same as the OSI model for networking, with most safety applications being in the application layer, meaning that this is where the safety monitor works, so to secure against an attack on this, it would be worth securing this layer.

The sensors can be attacked by spoofing, jamming, tampering, adversarial and DoS attacks. For communication attacks, these attacks would be sybil, replay, relay, malware, impersonation, MitM, black-hole and false information. The actuators on the other hand are vulnerable to sensor fusion attacks (i.e. by poisoning the machine learning models being used), piggybacking (i.e. rogue updates and false software) and supply-chain attacks (i.e. interfering with third party elements produced by others). Most solutions will involve either tradition methods such as introducing rules to the system, or to have AI to be more adaptive when it comes to security.

The authors of (Riffie, Riffie and Youssef, 2024) discuss the risks of connect vehicles and the integration of communications into their systems. The issue is addressed with potential solutions and challenges that exist for connected vehicles, which are the base of CAVs, as this is before they are even automated, and these risks may grow into bigger ones once automation is added. A well-known form of attack is DoS, which can be executed against various parts of the vehicle network, including communications and the CAN bus. An example of a real attack is the 2013 Jeep Cherokee, where poor Wi-Fi password generation allowed it to be bruteforced and used as an access point to access the vehicle and eventually make it to the unsecured CAN bus, where commands could be sent to all different ECUs connected to it in the in-vehicle network. While encryption is proposed as a large solution to many of these issues, as it introduces high level security at various levels, the emergence of quantum computing is a sincere concern as it grows as the current security methods may be too weak for it.

In (Nayak *et al.*, 2022) it was found that in VANET, there is a lot of allocation of resources, security and privacy needed. Unfortunately, this means there are a lot of open issues as there is a lot of computational power required to authenticate and validate messages to stop violation of the integrity property. The presence of trusted authority (TA) and RSUs are typically required for a secure infrastructure, but these are not always available.

In (Mostaq Hossain *et al.*, 2023), a survey of the key attacks on CAV systems are outlined. The connected and autonomous elements of CAVs cause serious security concerns for the future, especially in relation to user confidentiality and the integrity of messages being communicated between different vehicle components such as ECUs. Attacks on autonomous vehicles are grouped into the following:

- VANET: Attacks on internal and external networks.
- Sensors: GPS, IMU, LiDAR and Camera.
- Hardware: OBD Port, ECU firmware, Rogue updates (physical and remote).

For the VANET, the internal network attacks involve cracking passcodes and keys of the vehicle system, while performing V2X attacks or DDoS attacks are the remote elements. For sensors, spoofing and jamming are the key attacks to mislead the vehicle into potentially dangerous behaviour. The hardware attacks include accessing the OBD port to try and get into the CAN this way, physically upgrading firmware in the ECU to change behaviour or to implement rogue updates in the CAV.

In (Gupta, Maple and Passerone, 2023), the security in CAVs is investigated, looking at potential solutions. The paper suggests that support systems must be put in place (i.e. infrastructure) as CAVs cannot work in isolation, which would be done by introducing and ecosystem by introducing CAVs as edge devices, adding RSUs and using cloud computing as the backbone of the infrastructure. The attacks on the system can be software, hardware and network based, so these are the areas that must be secured. The crucial systems and functions in the CAV are found to be as follows:

1. Powertrain Controller
2. Chassis Safety Controller
3. Body Electronic Controller
4. Infotainment System
5. Navigation System
6. Sensor Systems
7. Network Systems
8. Power Management

These systems will manage different part of the CAV, working with the different internal controls and couple with AI to work effectively. The cloud component also has its own levels of management, and it is suggested that by securing various parts of the crucial systems will help mitigate the effect of attacks.

The threats to security in CAVs are bound to have an impact on safety, as discussed in (Gozubuyuk *et al.*, 2023). The key areas at risk as specified by this paper are sensors, control systems, connectivity, privacy, and access control. Sensor related attacks are all about spoofing or jamming the sensors, which will cause the perception to malfunction. The other serious attack is the in-vehicle network, which involves either hacking the cloud to get to the telematics unit or going through the infrastructure to get to a vehicle or direct V2V to get to the OBU to access the CAN bus to exploit the lack of security.

Analysis

The authors of (Jafarnejad *et al.*, 2015) address the increasing connectivity in vehicles resulting in an increase in potential attacks, particularly against their control systems. Attack scenarios and looked at, with changes being made to vehicle behaviour using an “Open Vehicle Monitoring System” (OVMS). The OVMS allows remote access to get information on the vehicle, which can work for different vehicles, such as the Renault Twizy used in the testing by the authors. The experimental setup consisted of the vehicle, which communicated

through OVMS, connected to communication methods such as OBD-II, GPS, and GSM (2G cellular). This was then linked to an OVMS server which could communicate with an application that was installed on an Android phone. The motor control ECU of the vehicle in this case was a Sevcon Gen4 controller. The port was found to be usable as a means of configuring the device (i.e., ECUs), so by accessing it and forging data frames it is possible to gain some level of control. A brute force attack was used to crack the passcode securing the access levels to different ECUs in the vehicle in mere hours, allowing changes such as throttle and even gear states. Various attacks could all then be executed remotely via Android application, including the following possible attacks (Jafarnejad *et al.*, 2015):

1. Changing vehicle direction.
2. Changing vehicle speed.
3. Setting unsafe parameters for the motor and voltages.
4. Changing motor direction.
5. Displaying false data on the dashboard.
6. Changing throttle input functions.

All these attacks could prove hazardous as they change the regular behaviour of the vehicle on the road, so I could model these in the presence of hazards to observe the extent of their effect on safety, though false data on the dashboard would mostly affect the driver reaction so this area may not be in the scope of my project unless I intend to model driver reactions as well. Solutions suggested included cryptography with the use of a hardware security module, which would be an ECU designed specifically to implement security as opposed to applying it to each ECU, meaning that overhead would not be as much of an issue. In some cases a limiting method could be applied so that only a certain number of password attempts are allowed so that brute force is rendered less effective (Jafarnejad *et al.*, 2015). IDS can also be implemented, mostly based on detecting unusual or anomalous behaviour in terms of the traffic in the in-vehicle network, as well as the possibility of monitoring the state of ECUs and alerting the driver to potential attacks. Protection mechanisms were found to be an important part of the problem so any models for security must have defences factored in.

(Jonnalagadda and Jakkala, 2020) are of the view that security analysis is needed for modern connected vehicles, with new attacks being introduced such as malware which could result in real-life safety threats. The authors here look through literature and experimental data to determine the knowledge gaps and potential solutions to vehicle security issues. A stated vulnerable area is the MirrorLink protocol used by vehicle makers such as Toyota to project the mobile screen and its applications onto the vehicle screen. The vehicle is found to be connected to infrastructure, mobile devices, other vehicles, the home, and even industrial systems, meaning that these could all potentially be used as access points to compromise vehicle systems through the connections and communications. The authors found that attacks used to access the vehicle through ports, media (i.e., C.D.), cellular or wireless access can all be used to gain full control of a vehicle, with remote attacks being more costly to the attacker. The range of attacks, including those across communications make it difficult to deploy vehicular networks, meaning that better solutions such as cryptography, firewalls or special security protocols may be needed at the communications to implement them safely and securely. Therefore, models should include attacks and defences against the vehicle at these

points on the attack surface such as communications to see the implications of these attacks when simulated.

(Koscher *et al.*, 2010) indicate a more practical approach, looking at vehicle security from an experimental view, something which is found to be very scarce in vehicle security. Their experiments show that if an attacker can access any ECU, they are able to use this to manipulate a wide range of safety-critical systems. The experiments are in-lab and on the road, with different attacks showing the ability to control distinct functions, such as the brakes and engine. The authors intentionally do not specify a particular threat model, looking at what a successful attack could then result in. An attack will either be done physically, meaning direct access to the car (i.e., through OBD-II port or install malware when briefly connected to the vehicle), or to use the wireless interfaces (i.e., Wi-Fi or radio) to compromise ECUs remotely. Experiments were performed on the extracted hardware from the vehicle (i.e., the CAN bus and ECUs), a stationary vehicle and even on the road. To better analyse and perform attacks, CarShark was used to sniff and inject packets into the in-vehicle network, as well as to perform the full range of attacks in these experiments. Packets injected included those to give false readings on the dash, disabling various lights, braking and even disabling the engine successfully, which has major implications on the road (Koscher *et al.*, 2010). It was found that safety-critical ECUs could be directly controlled, as well as loading rogue code followed by erasing all traces of the actions leading up to the attack, meaning the attack itself could be hard to find and prove. It is found that based on what can be done it is possible for vehicles to tolerate failure but not attacks, meaning that stronger measures such as control over what ECUs can do (i.e., commands they send) need to be in place so that the compromise of a single ECU does not lead to this catastrophic level of control and risk to safety.

(Malik and Sun, 2020) review some common cyberattacks against vehicles, going on to model them and identify which threats are most significant, with a simulation of these attacks to see their impact and significance. Different simulation tools are mentioned such as TransModeler and SimDriver, with the mention that not much work has been done on the security aspect of connect vehicles. Risk evaluation by the authors was modelled as the likelihood, impact, and risk of a particular threat to determine the priority and need for countermeasures. The open-source simulator CARLA was used for the experiments. It was shown that infecting one vehicle and propagating an attack to an entire fleet of vehicles could lead to thousands of fatalities, both vehicle and pedestrian. Through drive-by downloads (i.e., installing malware onto a device connected to the vehicle) it was also possible to disrupt the vehicle functions and lead to crashes (i.e., veering off lane). Attacks along these lines also included rogue mobile applications to take control of the vehicle and even organized terrorist attacks using vehicles which was shown to pose a severe threat to life (Malik and Sun, 2020). This strongly suggests that defences must be implemented in models and see their effectiveness in the presence of similar attacks.

In order to better understand the implications of attacks and defences, (Netkachov, Popov and Salako, 2019) look at modelling specific cyberattacks using very abstract models, with Monte Carlo simulations to show the losses due to any cyberattack, despite how the specific attack details are not taken into account, which could be very helpful in my project, especially against attacks that not yet known to us. The benefits of defence-in-depth, a layered defence method, are also explored in the presence of cyberattacks, all of which are modelled for a

power transmission system which includes workstations and substations that are all connected through different local area networks (LANs). A point of importance highlighted is the authors' interest in design diversity, meaning that various parts of the devices all have a range of different defences, including two protected channels and fault tolerance. The diversification of two channels means they are less likely to both experience the same failure or attack therefore maintaining their availability. Each channel has its own rates of failure and repair, with device failure only being true if both channels fail for a device. An attack that occurs successfully puts the devices in the system into a "Compromised" state which the device could be more likely to fail than in the "OK" state. All attacks are modelled with SAN, limited to a single attack rather than multiple attacks at once, which I would also need to consider the limitations of if attempting similar work (Netkachov, Popov and Salako, 2019). Each substation or device in the system is assumed to have its own firewall, as well as intrusion detection (IDS) deployed for each LAN. The attack could also either aim to compromise a single channel (independent) or both at once (synchronized attack). While it is assumed that for an independent attack the adversary is unaware of the defences, they are said to have detailed knowledge when performing synchronized attacks, making the risk of both channels being compromised must greater. The attack is said to either change data (i.e., threshold protecting an electrical device) or to increase the failure rate (in this case one thousand times higher). The authors then consider that there are maintenance inspections occurring at fixed rates which return the system to the "OK" state, as well as attack rates. Attacks and inspections are daily, weekly, monthly, or yearly, but it is also possible to have no inspections; these are all modelled in the experiments as the attacks that change protection thresholds or those that reduce reliability, looking at inspection and attack rates combined for different possibilities. The rate of inspections is shown to affect power loss, specifically lower frequency of inspection resulting in greater power loss due to more time between intervals where the system returns to the "OK" state. For monthly inspections in the presence of weakly attacks, the losses are 2% or less compared to the baseline of 98.45% loss in the presence of no attacks. It is stated that parameterization can be carried out using sensitivity analysis in practice. Modelling defence in depth for one or both channels of the device with different inspection rates shows the following (Netkachov, Popov and Salako, 2019):

1. For inspection rates of zero, independent attacks on any power device have much lower losses than for synchronized attacks compared to the baseline. Even yearly inspections are sufficient at this point for negligible losses.
2. Synchronized attacks required much higher inspection rates, with monthly inspections being the minimum requirement to mitigate most losses caused by device failure from this type of attack, shown to happen around 7.5 hours after the attack.

It is seen that if an abstract model is given decent parameters, then expected losses (i.e., failure and power loss) can be established without knowing the attack itself provided the parameters are well set. Also, implementing defences while considering the knowledge level of the adversary it is possible to look at solutions such as defence-in-depth and their effectiveness. Therefore, it is worth having my own abstract model for CAVs and the relevant subsystems to see what losses (i.e., integrity and availability) I can expect from specific attacks.

A more recent paper by (Cao *et al.*, 2022) suggests that it is possible to exploit the presence of LiDAR sensors in vehicles to essentially “delete” a pedestrian or similar moving objects from the view of the perception system using a laser, as part of a newly suggested group of attacks called physical removal attacks. By spoofing LiDAR data gathered from the laser and feeding it back to the sensor, it is possible to attack the sensor up to 45 degrees with the laser, removing 90% of the visible data points scanned by the sensor with 92.7% success rate (Cao *et al.*, 2022). The injection of the echoes of data (i.e., reflected surrounding signals) causes legitimate information in the cloud points (i.e., the data points scanned by the LiDAR) to be discarded in favour of spoofed ones. The attack is still effective in the presence of combined camera and LiDAR, meaning sensor fusion is not as effective as it would be against regular spoofing in this case, with increased risk of collision, as well as being even more effective at night (Cao *et al.*, 2022). While this attack does sound interesting, it seems like spoofing so a new attack model would not necessarily be required when evaluating the resilience of sensors in this case.

(Khan, Khan and Arshad, 2023) proposed that a method for cybersecurity to increase trust with users is the use of digital labels to display the level of cybersecurity and measures in use in a vehicle. This lack of trust is made clear in how almost 75% of the US population are fearful of even being in a self-driving car in the first place. Labels are like energy labels or otherwise that provide assurance to the users, except in this case it works by having a system called CSBox get the information on cybersecurity status from the vehicle, which returns data that is then used by CSBox to generate the cybersecurity health report of the CAV. However, the issue seen here is that to provide assurance, the vehicle must in fact be secure and the security data retrieved must be valid. For example, if an attack is launched that compromises the integrity of the vehicle, specifically CSBox, false information could be provided to the drivers giving them a false sense of security, so there must be a clear, transparent process and good security in place to make sure the display from CSBox cannot be tampered with.

The paper by (Girdhar *et al.*, 2022) looked at the modelling of threats against cyber-physical systems, specifically CAVs, with a detection system proposed to identify unusual behaviour during a crash scenario. Malfunctions and cybersecurity issues were found to be the most serious areas affecting vehicle safety. A table constructed by the authors based on SAE J3016 (i.e., autonomy levels) showed that from level 4 onwards is where the vehicle is truly a CAV, requiring communications as well as a centralized architecture. The sensors at these levels include radar, sonar, camera, LiDAR, IMU and GPS, giving an idea of what to consider when modelling a CAV compared to a regular vehicle.

Scenarios for attack used in the paper involved attacking infrastructure to compromise integrity to send incorrect information to the vehicle, or to infect the system and corrupt components such as sensors and ECUs. Abnormal behaviour in scenarios due to attack that were shown included (Girdhar *et al.*, 2022):

1. Sending of an incorrect signal (i.e., green) to the CAV so that it speeds up despite having detected a vehicle (i.e., continues through an intersection when it should stop).
2. False data sent through compromised ECUs so that ADS malfunctions, resulting in speeding up after a pedestrian is detected.

The authors of (Culley *et al.*, 2020) look at system design for an AV, with the object detection (i.e. perception) being 25FPS and a precision of 92%. The vehicle systems are commonly broken down into perception, decision, control, and actuation, giving us an idea of what kind of structure a basic vehicle model should have, with perception being an area that requires more research. The modelled system includes various sensors such as cameras, LiDAR, GPS and IMUs. Results showed 25FPS for the perception, 87% detection accuracy and 93% detection of objects (in this case cones) that are in the frame. These statistics have potential to be incorporated into any model of the vehicle, especially the FPS which is the rate at which frames are received by perception.

In (Ramsbrock, Berthier and Cukier, 2007) an experiment was conducted in which 4 computers were set up and left connected to the internet to see how often they would be attacked. The profile of attackers was determined by looking at specific actions taken by the attackers such as changing configuration, changing passwords, and running rogue code. The attackers in this case were compromising SSH on Linux computers, aiming to crack the password to gain access to the system. The results of this paper indicated that each computer experienced an average of 2805 attacks daily, but with very few actions taken. This suggested that that attacker was using automated scripts with low skill so they did not have time to take advantage of compromised hosts, which means the behaviour of the attacker is different to how it would be against a CAV if successful. However, the primary interest here is the rate of attacks which could be used in attack models against CAVs, as they are on-road computers being attacked frequently.

A means of testing safety and security called VeriCAV was developed by (Levermore and Peters, 2020). This places the system under test using simulations to assess the effectiveness of an autonomous driving system. It is indicated that the system should be robust and modular, with a reliable amount of testing done, as it is not possible to test large amounts.

The authors in (Sun *et al.*, 2024) propose the use of secret keys across all communications as a means of securing the communications channels between vehicles as well as their wireless networks. This aims to mitigate the issue of authentication as the integrity problem is one of the most serious when it comes to safety in autonomous vehicles.

The authors of (Peng *et al.*, 2024) outlines the idea of autonomous ground vehicles (e.g. cars), which have a range of sensors and similar components that can be vulnerable to different attacks on different elements such as the sensors, which can be very stealthy to avoid detection, and this paper looks at the vehicle under such circumstances. The stealthy attacks on sensors are found to alter the trajectory into a dangerous zone while not affecting the internal states of the dynamic lateral controls. There are still other attacks that could be explored, especially those against communication channels (i.e. V2V, V2I, V2X), as well as DoS and false data injection, and the idea of how to defend against all of these.

Datasets

The authors of (Iqbal *et al.*, 2022) chose to try and simulate the entire vehicle network when subject to attack, providing a public dataset. The simulations are done using the Eclipse MOSAIC model, modelling V2V and V2I scenarios, for which replay attacks and false information attacks are performed. As V2V and V2I can be considered part of V2X, all messages created in the simulation are V2X messages, with DENM (Decentralized

Environment Notification Message) used to indicate accidents, while CAM (Corporate Awareness Messages) are used to perform V2V and V2I in attack scenarios.

The replay attack in this case takes the signal from an emergency vehicle that typically causes CAVs to stop and make way. By doing so the replay can be used to forced other vehicles to stop provided they receive the signal to make way for another vehicle exploiting this to gain priority over others when using the road. The false information attack in this paper involves communication to other vehicles information about a traffic incident that is not there so that they will take an alternative route. Therefore, the results are that vehicles in these cases will reach their destination later than the malicious vehicle (Iqbal *et al.*, 2022).

Detection methods proposed by other authors to avoid various attacks used ML models that observe the vehicle position, range of acceptable values, signals strength and the packets transmitted (Iqbal *et al.*, 2022). The attacks in this paper merely altered vehicle course and caused delay but could have been much more severe.

The VeReMi (Vehicular Reference Misbehaviour) dataset by (van der Heijden, Lukaseder and Kargl, 2018) is for vehicle misbehaviour, including incorrect or malicious messages. The authors claim that this is the first dataset of its kind and aims to provide data due to it being hard to constantly build detection mechanisms for every attack and test them. Dataset cannot replace detailed analysis, but this replaces more time-intensive methods which are prone to error. Attack probabilities that could be used in attack models are 0.1, 0.2 and 0.3, for distinct types of attack which in this paper are for falsified positioning (i.e., interfering with the GPS). Attacks of this type can be (van der Heijden, Lukaseder and Kargl, 2018):

1. **Constant:** Using fixed false position.
2. **Constant Offset:** Fixed offset of the current position of the vehicle (i.e., nearby but misleading by adding to the current position data).
3. **Random:** Picking random false positions to give the vehicle.
4. **Random Offset:** Random offsets given to the vehicle to mislead it within a predefined fixed area.
5. Changing vehicle messages in patterns over time.

This dataset should be considered non-interactive, so it is not possible to change it or vary parameters to get new results, and it may be possible to attack the fusion system to exploit small errors in the system. False positions and random false offsets aim to just transmit erroneous positions, while the other types of attacks aim to confuse applications. Therefore, as this is for false positions, which would be the attack for which this data will be most useful in modelling.

Later on, the VeReMi dataset was extended by (Kamel *et al.*, 2020) by introducing new attacks that allow manipulation of message frequency, message contents and digital certificates. Suggested solutions include detectors for plausibility and for fusion. Errors are also introduced to the sensors, such as position, velocity, acceleration, and the vehicle heading. Attacks include DoS, replay and Sybil. Malfunctions introduced will be based off position, speed, delays, and message types.

Vehicle Standards

An important part of correctly building safe and secure CAVs is to have regulatory bodies as well as good standards for all manufacturers to abide by. Some of the most important ones so far are:

7. **ISO 26262:** The main standard for the functional safety of all vehicles, with guidelines and vocabulary.
8. **ISO 21448:** Safety of any function being incorporated into a vehicle, as each functionality has its own issues to consider.
9. **ISO 21434:** The main and most recent cybersecurity standard when it comes to vehicle design, including the guidelines, governance, and necessary threat analysis.
10. **SAE J3016:** The list of taxonomies and definitions in relation to the vehicle, driver as well as developing various regulations and specifications. This standard also contains all the levels of vehicle autonomy and their details.
11. **PAS 1885:** A standard for basic principles of vehicle cybersecurity, regarding how organizations should be dealing with cybersecurity in terms of understanding, culture, and the overall cyber-landscape.
12. **BS 31111:** Like PAS 1885 but created for managing cyber-risk and resilience for governing bodies and executive management.

Definitions and some descriptions are given by (Sanders and Meyer, 2001), describing SAN as a method of measuring performability used since the 1980s. The SAN can be used to look at features of a model such as performance and dependability, as well as being used to represent the behaviour of various systems with time constraints. These can be used as extensions to Petri nets and can be used to model a range of scenarios and systems including fault tolerance and system degradation, making them especially useful for modelling attacks and defences in my case. Effective models must have as little ambiguity as possible while having a sufficient level of abstraction (i.e., remaining sufficiently generic) to allow a system to be represented easily but also in a formal way to allow reliable results to be obtained. Activity networks can be considered generic Petri nets consisting of timed activities for state transition, places to describe states, as well as input and output gates that can check requirements or assign values, respectively. All input or output gates are connected to an activity, as well as to one unique place, but it is possible to connect different gates from an activity to a single place. When using SANs, all probabilistic behaviour (i.e., parameters and mathematics) must be made clear both in terms of variables and all equations relating to the statistical models of any activities. The timed activities of SANs start at activation and end at completion for the duration of the timed activity (i.e., the enabled period).

The NIST special paper by (Ross, McEvilly and Carrier Oren, 2016) consisted of different considerations for security, starting with fundamentals such as capabilities and dealing with various faults, as well as the strategy for security. The life cycle of the system must also be considered in terms of security, including management, design, analysis, and disposal (i.e., end of life cycle). An extremely useful element was the security policy, consisting of the confidentiality, integrity, and availability (CIA) triad of the different elements of the system.

BS 31111 (BSI, 2018a) looks at resilience and risk management, which requires adequate resources to meet the objectives in their policy. Security arrangements currently do not carefully consider factors that can adversely affect resilience, which can lead to projects

being implemented that consider the benefits of the system without looking at current or emerging threats. A resilient organization must have strong governance, use of resilience on their designs and on the organizational level. There must also be continuous and strong cybersecurity, with the ability to learn and change from changing situations, while also sharing information effectively. It is also important that there is a cyber-aware culture, to influence human behaviour in a way to limit breaches due to human error which may be due to a lack of cybersecurity knowledge. There are many different rules and regulations that must be followed, including the ability to adapt, transforming business, risk management, and correct response. Communication is key in all of these, as all the involved groups need to be able to make their positions and relevant issues known.

The ISO 26262 standard (BSI, 2018b) is intended for functional safety, namely for different devices produced, such as those in the automotive sector. It aims to create a safety culture within an organization, effectively promoting and communicating functional safety, while also managing the quality of system safety. The standard consists of vocabulary, management, phases (i.e., concept, where risks and hazards are determined), product development and safety analysis (i.e., ASIL safety levels for devices). There are also additional guidelines on how to apply the standard, not just for cars but for other vehicles such as motorcycles.

PAS 1885 (BSI, 2018c) is for how an organization should deal with venturing into the space of vehicle security, looking at resilience and risk management. Existing standards must be met, as well as looking at cyber risks, levels of preparation and prevention. It is also important to look at putting in the correct amounts of resources, with strong governance of cybersecurity and awareness of the changing cyber-landscape.

The purpose of the ISO 21434 (BSI, 2021) standard is to address cybersecurity in E/E systems and architecture. It is important to have cybersecurity policies and processes, manage risk and foster a culture of security within any organization that considers cybersecurity important. General considerations are the overall cybersecurity, such as what someone could do to try and cause problems, but in no way a specific idea, this is basic and more of a starting point. Cybersecurity should be regularly audited to check for compliance, with good planning and making sure all responsibilities are clear when it comes to security.

Cybersecurity must be regularly monitored, with regular evaluation for the life cycle of the product or system, with regular analysis and management for any vulnerabilities that emerge. Threat analysis and risk assessment methods include identifying threat scenarios, rating their impact (i.e., how dangerous), looking at the potential attack path taken by the adversary and the feasibility of the attack. After this the level of risk can be determined so an organization knows how seriously to treat a risk.

The code of practice for safety and cybersecurity by (IET, 2021) aims to not only hold organizations accountable for issues in these areas but to help them better manage these problems. The principle behind safety and security in digital systems which are connected to physical systems are that if it is not secure it cannot be considered safe, as there is no solid line between the safety and security of digital systems and control systems, meaning there is a lot of overlap and therefore a lot of safety implications. There is a lack of a current common method of approaching safety and security, with different groups of people such as IT experts or engineers seeing them in different ways and therefore placing priority on different areas, so

it would help a lot if there was a consensus of what exactly is the most important when it comes to these two areas, especially when they overlap. Safety and security must be an active part of all processes, being taken into consideration when considering risks, while implementing methods to keep architectures resilient but with solutions being proportional to the level of risk. Considerations for safety and security need to be system-wide, with assessments performed to determine the best solutions to any risks.

Solutions to the possibility of faults occurring for a system and its components can be grouped as follows (IET, 2021):

5. **Fault avoidance:** Create the system with no faults in the first place so there is little to no future correction to be made.
6. **Fault removal:** Faults must be found and removed through the appropriate inspection methods that are of high quality.
7. **Fault tolerance:** The system should be designed in a resilient manner to deal with faults and remain available even when encountering some, avoiding any total failure.
8. **Fault warnings:** There must be methods of alert for faults as well as giving time for the system to recover properly from such issues.

It should be noted that this code of practice is intended for the engineers involved and those who manage these systems, so that they can use these good practices when making decisions regarding their systems.

In J3016, the vehicle standard aims to provide information into the recommended practice of a vehicle, covering the levels of autonomy for vehicles (SAE, 2021):

- **Level 0:** No driver assistance, meaning the driver is entirely responsible for the vehicle and its motion.
- **Level 1:** Driver assistance, where parts of the DDT (Dynamic Driving Task) are done by the vehicle, but the remainder must be performed by the driver, and the fall-back is still the driver.
- **Level 2:** There is partial driving automation, so the system performs motion control, but the driver is still the fall-back, with a limited ODD (Operational Design Domain, meaning the constraints under which the vehicle works correctly).
- **Level 3:** In conditional driving automation, the system is still in charge of the DDT, but now object and event detection and response are done by the system, with the driver being a fall-back from the DDT, with limited ODD.
- **Level 4:** High driving automation places all tasks in the hands of the system, with no driver fall-back, but the ODD is still limited, so it will only work in specific areas (i.e., with geofencing).
- **Level 5:** Full automation is the desired result, with the system doing everything without any driver input or exceptional circumstances, and no ODD limitation.

Driver automation is done by performing dynamic driving tasks (i.e., during motion, with changes) and with potential for object event detection and response (i.e., see an obstacle, respond by avoiding or stopping), and having a fall-back, such as in the case where the vehicle cannot handle it. The range of motion is also important, as it may be controlled both laterally and longitudinally, with some aspects given to the driver in lower levels of automation (SAE, 2021).

(BSI, 2022) makes it clear that safety is implemented at separate phases, such as in the design phase, verification phase, and validation phase, so that performance can be checked by different means including testing and technical reviews, and simulations. To improve safety of intended functionality, it is advised to have the full system specification, as well as the identification and evaluation of hazards (i.e., those caused by a functionality). Modifications should be made where needed to reduce risk, with verification and validation of safety. For safety in automated systems, I must make assumptions to effectively parametrize the model as much as possible, especially where I do not know all the information.

Here the authors of the IEEE standard state the minimum assumptions that need to be made about any automated system when developing a safety model. Some important terms include (IEEE Standards Association, 2022):

7. Acceptable risk: Sufficient level of safety, as there will always be some risk, but this needs to be within a certain bound to pose the least threat possible.
8. Event: Any changes to the driving system or its environment.
9. Formal models, which are designed mathematically or otherwise, for simulation, of the system in question to describe its behaviour.
10. Representation: All the information about a system in a particular case or scenario which can allow its effective description.
11. Safety envelope: The limits and conditions under which the system operates to ensure safety.
12. Safety-related Model: The model of the safety system to show all aspects of safety as well as the potential failings of such a system.

Any safety-relevant object such as other vehicles that could play a role in safety in the form of an event that could potentially be hazardous. Anyone who is not a vehicle user (i.e., a pedestrian or other non-driver) can be considered a vulnerable road user (VRU) in these cases. There are only some cases that assumptions can be made for, specifically scenarios such as (IEEE Standards Association, 2022):

6. Driving next to other road users.
7. Driving behind other road users.
8. Driving in between road users of varying speeds.
9. Interactions with VRUs on the road (i.e., how to deal with a pedestrian crossing).
10. Road users that may or may not be visible (i.e., obstructed by objects or poor weather).

In all these cases every element that is not the fixed road environment (i.e., scenery) is considered a dynamic entity and therefore any assumptions made are not always true as the environment with dynamic users will vary, generating hazards at any given time.

Assumptions made for modelling assume that the static environment will not change in any way, so my focus would be the dynamic users such as vehicles or pedestrians.

An example of an autonomous vehicle does not always have to be a car, as an existing example that may also be deployed is the driverless bus. In (Myklebust, Stålhane and Jenssen, 2023), a trust case is shown, as well as different security cases. The trust case here is where the bus is driverless, but its operator is moved to a remote-control room where they can take control if anything goes wrong. Another essential element is the standards for safety

and security, one of the most recent and relevant being the ISO 21434:2021 which specifies the cybersecurity risk management requirements for vehicle electronic and electrical system throughout the entire vehicle life cycle. The standard includes requirements for manufacturers making components for parts of autonomous vehicles. Some AI standards are also required as a lot of ML algorithms are not always transparent so explaining them is difficult, meaning that there must be some sort of standards of the transparency of AI and what it does in CAVs.

Modelling

While not related to the background of CAV, this project will involve having to model the behaviour of a vehicle in the presence of various attacks to effectively evaluate cyber-resilience. As many behaviours and attacks against CAVs cannot be known or predicted fully, probabilistic modelling, simulations and potential other models will have to be used to get the closest approximation to the expected on-road behaviour of a CAV.

Stochastic Activity Network (SAN)

SAN models, also known as stochastic activity networks have been used in performance models for decades, even being used in software such as Mobius, which is currently being used in this project to model CAV systems to investigate their cyber-resilience, so that I can see how the system behaves under different circumstances (Sanders and Meyer, 2001).

The SAN model is said to consist of the following components for its ‘Atomic’ components, namely for a single network not yet joined to any others (Sanders and Meyer, 2001):

- **Place:** These are circular objects in the network that are assigned a value to indicate a state, such as compromised or failed. The system usually transitions between these when being simulated.
- **Timed Activity:** An activity that happens over time, set to perform a task at a fixed rate according to whatever distribution is chosen for it, such as exponential distribution.
- **Instantaneous Activity:** If an event happens on a state is reached connected to this, the activity happens immediately.
- **Input Gate:** This is a gate that the system passes through, with a predicate/requirement has based off existing states or parameters in the system which must be met to pass through this gate, otherwise the next state or event after this gate will not be reached.
- **Output Gate:** This is usually connected to another activity and values are output or set here before continuing to the other parts of the activity network.
- **Extended Place:** A more complex form of place which can be used for values such as arrays or structures, which are very good in a complex system.

These components are linked together via connections that can be curved or straight, with different cases occurring at activities, like if or case statements in programming. Various devices can be modelled this way, as well as being able to model Markov chains (Popov, 2015). Once the ‘Atomic’ component is created this can then be linked to other components, so that a place can exist in two different models, allowing another model, such as an attack, to affect another system. This can then be parameterized for numerous iterations and simulated either via numeric solver or Monte Carlo simulation.

ADVISE

One of the formalisms supported in Mobius is ADVISE, which allows for the creation of very specific attack models in the form of adversary profiles. This includes the following (Sanders and Applequist, 2012a):

6. Attack Steps: Actions taken by the adversary, with each step described by inputs, outputs, and the properties of the action itself.
7. Access: Areas the adversary has access to or may eventually gain access to. This consists of variables which store the access states, so it is known if the adversary has access to certain areas or not.
8. Knowledge: What kind of knowledge is available now or in future to an adversary, such as passwords to a certain area of the system which could allow for more effective attacks.
9. Skill: The capability of the adversary in different areas, which are stored as different variables indicating how strong the adversary is in different areas.
10. Goal: The aim and motivation of the adversary, which can range from stealing information to trying to shut down the entire system.

The ADVISE attack models are created and combined to form an attack execution graph, which can then be assigned different information such as the cost of an attack (i.e. the damage), preconditions, execution times (i.e. time intervals between attack steps), and the different outcomes from an adversary.

The authors in (Ford *et al.*, 2013) look at the implementation of ADVISE formalism in the Mobius tool. The formalism creates an attack model that contains all the adversary properties which can then be converted into a single executable event model. The attack model contains multiple state variables that define the access, skill, knowledge, and goals of the adversary in question. The state variables are assigned values as follows:

3. Boolean: The access, knowledge and goals are just flags, indicating if the attack does or does not have certain abilities (e.g. for access, does the adversary have root admin).
4. Skills: This is a measurement of proficiency, ranging from 0-1000.

The attacks steps are model elements that can cause the model to change state. These each contain a precondition, the relevant time distribution, and the cost of an attack. Each step is assigned its own outcomes which include the probability of the outcome, probability of detection and the probability of different effects.

The attack execution graph should not be confused with the adversary profile, which is what feeds the necessary parameters into the graph itself to define the adversary with key elements (i.e. the values of the skill, knowledge, etc.).

The attack model itself has a decision stage and an action stage. The decision stage looks at the different attack steps and outcomes that may happen in future and uses these to choose the most attractive next step for an adversary, and then the action is taken to perform the selected steps.

When using the solver for the state spaces to be simulated in ADVISE, it must be noted that not all models in Mobius can be transformed into Markov chains, which is especially true for ADVISE attack models as the adversarial decisions are not solely dependent on the current state. This is a limitation which makes this less suitable for modelling CAVs as Markov chains are a key element in the models and this not only removes this property in the state space but also may overcomplicate the model itself with more information being required about the adversary that I simply may not know, while using simple SAN modelling allows for more general adversaries and a higher level of abstraction.

(Mariotti *et al.*, 2023) propose an extension to the already existing ADVISE formalism into a greater level of abstraction by modelling the attack patterns using the CAPEC (Common Attack Pattern Enumerations and Classifications) database and adversaries profile using the TAL (Threat Agent Library) database.

The entries used from the CAPEC database describes an attack pattern as well as the following elements relevant to an attack:

6. Description of the attack.
7. Prerequisites to carry out the attack.
8. Resources required (e.g. devices and tools).
9. Skills required for the attack.
10. Consequences of the attack, including the scope and impact.

The goal of ADVISE Meta is to make it easier to model by removing the need to do so much modelling work by hand, consisting of the following parameters:

11. Component: Elements that could be part of the system.
12. Relationship: Could exist between components in the system.
13. Attribute: Characteristics of components which can be used as parameters for attack steps connected to any component.
14. Access: Level of access an adversary has at the start of the attack.
15. Skill: Capability of the adversary.
16. Knowledge: Information possessed by the adversary, in advance that may affect the attack.
17. Other State Variable(s): Other variables that could define adversary goals.
18. Attack Step: Defines steps of an attack that could be performed by an adversary.
19. Adversary: Defines the profile of an adversary with various characteristics.
20. Metric: Those such as goals are metrics, but more can be added.

The ADVISE Meta modelling involves adding all the elements into a system diagram and setting all the parameters and connecting components through their relationships based on system architecture.

The attack steps consist of the attack cost, execution time, preconditions, success probability, and detection probability.

The TAL database contains the following elements:

9. Access
10. Intent
11. Limits
12. Objective
13. Outcome
14. Resources
15. Skill Level
16. Visibility

It is possible to combine elements from both CAPEC and TAL with the ADVISE formalism to create the ADVISE Meta. Remaining work includes trying to integrate defences into the models, with current defences represented by the probability of a successful attack or other components such as the strength of existing defensive measures.

GSPN

In GSPN and Interval GSPN (IGSPN), the timed activities are all exponentially distributed regarding their firing times. GSPN expresses models as some of the following (Galdino and Maciel, 2006):

6. Finite set of places (i.e. how many states I can go to).
7. Finite set of transitions (i.e. the arcs or timed activities).
8. Inputs, outputs, and inhibitor functions.
9. Priority function to determine the importance of different transitions.
10. A function to associate the transitions with real values including the exponential parameters and the weights.

(Galdino and Maciel, 2006) investigated extending GSPN using an interval-based approach to create IGSPN. The aim of GSPN is to evaluate performance and reliability of petri net models, with IGSPN taking a closer look at the effects of variations in the exponential weights and transition rates. These are assessed by looking at various sources of error such as rounding off or truncation relative to the initial data, giving intervals that indicate the minima and maxima for each of these values.

What IGSPN does is that it adds an interval model to all these components of the GSPN, representing uncertainties that could exist in different model parameters. This suggests that while GSPN could be effective for reliable modelling, especially with the interval addition that could account for uncertainty, it is still restricted by the fact that all transitions are

exponentially distributed, which limits the kind of behaviour that could be modelled only to that of events which are known to be exponentially distributed.

PRISM

PRISM is modelling method which represents different models using its own unique code. The latest version at the time of writing introduced the quantitative checking of probabilistic timed automata as well as probabilistic, nondeterministic, and real-time characteristics.

PRISM provides model checking for various probabilistic models, including discrete and continuous-time Markov chains and decision processes, as well as providing a language to represent the models with. It can be used in a range of applications including wireless technology and quantum cryptography.

Both deterministic and nondeterministic models are supported for both discrete and continuous time, with the ability to reuse models and can be described as follows (Kwiatkowska, Norman and Parker, 2011):

7. Discrete-time Markov chains
8. Markov Decision Processes (MDP)
9. Probabilistic Automata
10. Continuous-time Markov chains
11. Probabilistic Timed Automata (PTA)
12. Priced Probabilistic Timed Automata

The overall functionality of PRISM includes the ability to model and build different models including probabilistic timed automata (priced and regular), with the ability to add costs and rewards to enhance them further. There are also different model checking methods which can be done with symbolic and explicit-state, with discrete-time simulation and the ability to import models.

Probabilistic timed automata are a form of state machine with a real value clock (variables with real values that increase over time) added to them, to make them like timed automata with probabilistic choices which can allow them to behave like Markov decision processes. The transitions in these automata can reset the clock when the state changes, with the transitions being probabilistic and non-deterministic. If the costs and rewards are accounted for in these models, the automata can become priced probabilistic timed automata.

Probabilistic timed automata are verified in the following ways:

3. Quantitative abstraction refinement uses probabilistic abstractions and analysis to refine at each step of the model
4. Digital clocks are integer valued and perform automatic translation of the model to its equivalent finite state automata and checks it using Markov decision process-based methods.

Additionally, the discrete-event simulation engine can be used to debug and check models, and optimal adversary generation is also provided based on the reward models, creating counterexamples to the model in the form of attacks.

Such a modelling tool has many possibilities, especially with all the verification methods available, but it is all written as a coding language with very little visualization, making it harder to see how it may reflect the architecture of the system.

Other

Despite SAN being the key model, and the project being focused on this type of modelling, there are still other models that need to be considered as these are usually used to evaluate risks and threats to various parts of systems or processes. Some of these models include:

STRIDE: Looking at Spoofing, Tampering, Repudiation, Information Disclosure, DoS, and Elevation of Privilege. These are key security elements as they embody what an attacker may aim to achieve, so by assessing these it is possible to see which properties are at risk of being violated for specific system parts (Fowler, 2020).

DREAD: Developed by Microsoft, this looks at how much damage an attack can do, how easily it is reproduced, the extent to which it can be exploited (i.e., difficulty), users affected by the attack and how easy it would be to discover the vulnerability in question (Fowler, 2020). Each part is given a score, and the average of these scores is taken to evaluate the overall risk.

PASTA: The Process for Attack Simulation and Threat Analysis consider threats from the business and technical perspectives. This approach is centred around an attacker and aims to product output that evaluates and enumerates threat scores (Fowler, 2020).

HEAVENS: Short for ‘Healing Vulnerabilities to Enhance Software Security & Safety’, this is a systematic approach of modelling, which aims to determine the security requirements as well as performing security testing, it uses case studies, threat analysis and risk assessment, which can then be used to better identify, classify and assess threats and/or risks (Fowler, 2020).

EVITA: E-Safety Vehicle Intrusion Protected Applications evaluates protection of security engineering, threats and security requirements, with the original European project having focused on protecting the in-vehicle network in the automotive domain (Fowler, 2020).

OCTAVE: Operationally Critical Threat, Asset and Vulnerability Evaluation assessment looks at information security, identifying the threat, the assets and evaluating the vulnerability, so that the assets most important to an organization can be better protected by seeing which threats or vulnerabilities put these assets in danger (Fowler, 2020).

FMEA: Failure Mode & Effect Analysis looks at individual functions or processes, the failure that has occurred and performs analysis of the effect. This includes the frequency, severity, and detection of such a failure, taking note of recommended actions as well as who should be taking these actions to prevent such failures in the future (Forrest, 2010).

FMVEA: This is much like FMEA except that vulnerabilities are now also considered (hence the ‘V’), meaning that threats and their effects are considered in parallel with failures when evaluating the function or process in question. Threats and their agents are analysed

repeatedly until there are none left, so this model now considers both safety and security (Schmittner, Ma and Smith, 2014; Schmittner *et al.*, 2015).

CHASSIS: Looking at assessing the potential harm to an information system, this also combined safety and security, both in terms of functional and safety/security requirements, while utilizing use cases. This type of model looks at users, functions, services, potential harms (as well as their scenarios), and mitigations that exist against harm in the current system (Schmittner *et al.*, 2015).

HAZOP: Hazard and Operability Study is for highly detailed system designs, ideally with multidisciplinary team that select a stage or section of the system/process, looking at each section like FMEA, so all hazards, implications and countermeasures are known (Crawley and Tyler, 2015).

Another note on models, specifically semi-Markov can be found in (Littlewood, 1979) looks at a modular program designed to transfer control between different modules. This is assumed to follow a semi-Markov process and is considered failure-prone with its processes being assumed to be Poisson for failures. The failure process of the program in question is said to be an asymptotic Poisson process. It is stated that two non-overlapping intervals of time could be considered asymptotically independent with their own blocks of behaviour. The contribution of this paper is therefore demonstrating that the time to an absorbing state for a semi-Markov model is asymptotically Exponential where the rate of an accident (i.e. absorbing state) is smaller than the rate of transition to the non-absorbing states, even being able to provide a formula for how this works.

For more details on any of these modelling types and how they work, the details can be found in Appendix 12.1 where they are fully explained with diagrams to show their processes where possible.

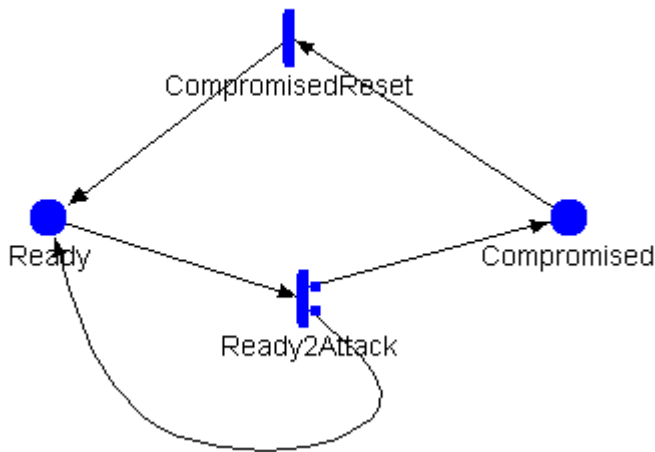
13 Appendix 2

This appendix provides a detailed description of all models used in the thesis. The descriptions were derived using the built-in functionality of Mobius to document the models.

Each description contains full details on the atomic models, the respective composed model, Reward model, the Study, and Solvers (if applicable).

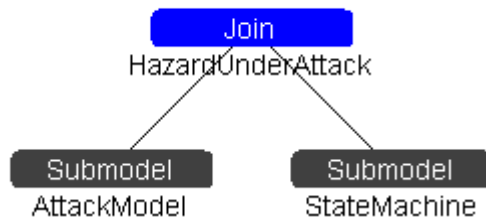
13.1 Separate Vulnerability Modes Model (Improved DESSERT'2022)

Model: AttackModel (Atomic Model)

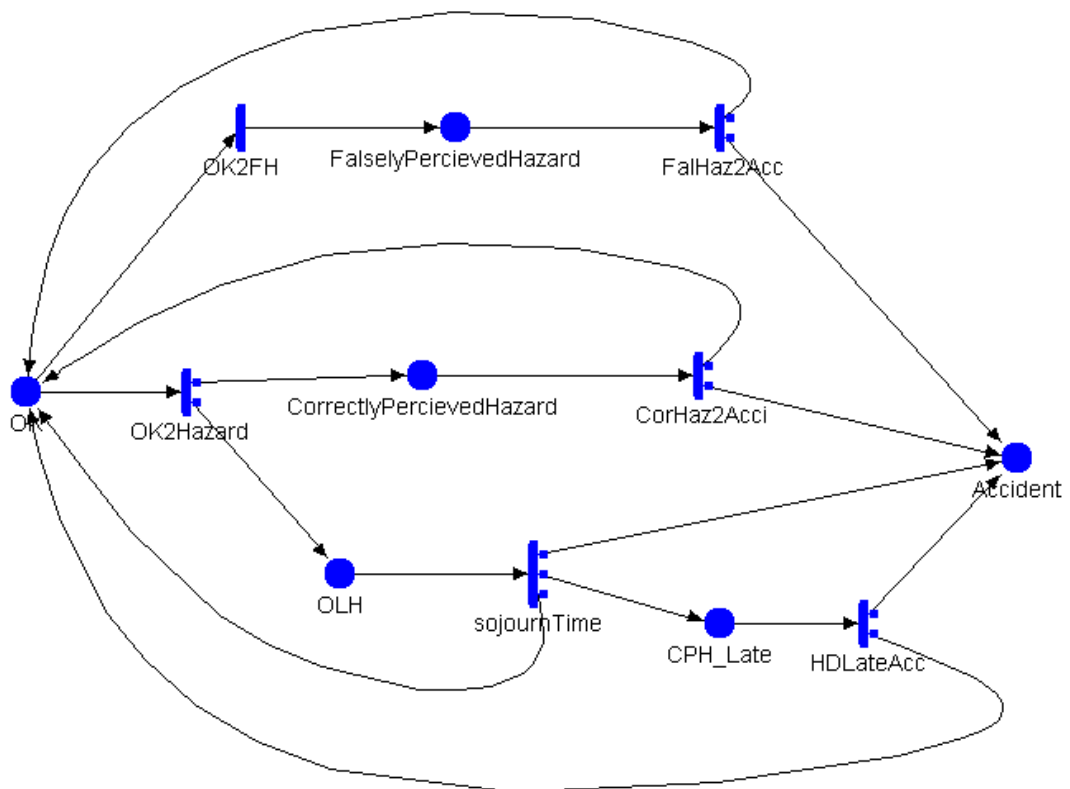


Place Attributes:	
Place Names	Initial Markings
Compromised	0
Ready	1
Timed Activity:	CompromisedReset
Distribution Parameters	Rate return(attackReset);
Activation Predicate	(none)
Reactivation Predicate	(none)
Timed Activity:	Ready2Attack
Distribution Parameters	Rate return(attackRate);
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 return(attackSuccess); case 2 return(1.0 - attackSuccess);

Model: Hazard_Attack (Composed Model)



Model: StateMachine (Atomic Model)



isRSSperfect - a boolean variable:

1: if RSS is perfect (detected hazard cannot lead to accident)

0: if RSS is not perfect, then the hazard can lead to a an accident.

 Compromised

Place Attributes:	
Place Names	Initial Markings
Accident	0
CPH_Late	0
Compromised	0
CorrectlyPercievedHazard	0
FalselyPercievedHazard	0
OK	1
OLH	0
Timed Activity:	CorHaz2Acci

Distribution Parameters	Rate return((CH2AccRate * pow(AccidentScale,Compromised->Mark()))*(1 - isRSSperfect) + CH2OKRate);
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 return(CH2OKRate/((CH2AccRate * pow(AccidentScale,Compromised->Mark()))*(1 - isRSSperfect) + CH2OKRate)); case 2 return((CH2AccRate * pow(AccidentScale,Compromised->Mark()))*(1 - isRSSperfect)/((CH2AccRate * pow(AccidentScale,Compromised->Mark()))*(1 - isRSSperfect) + CH2OKRate));
Timed Activity:	FalHaz2Acc
Distribution Parameters	Rate return(FH2AccRate + FH2OKRate);
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 return(FH2OKRate / (FH2AccRate * pow(AccidentScale,Compromised->Mark()) + FH2OKRate)); case 2 return(FH2AccRate * pow(AccidentScale,Compromised->Mark()) / (FH2AccRate * pow(AccidentScale,Compromised->Mark()) + FH2OKRate));
Timed Activity:	HDLateAcc
Distribution Parameters	Rate return((CH2AccRate * pow(LateCH2AccScale,Compromised->Mark()))*(1 - isRSSperfect) + CH2OKRate);
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 return((((CH2AccRate * pow(LateCH2AccScale,Compromised->Mark()))*(1 - isRSSperfect))/((CH2AccRate * pow(LateCH2AccScale,Compromised->Mark()))*(1 - isRSSperfect) + CH2OKRate)); case 2 return(CH2OKRate/((CH2AccRate * pow(LateCH2AccScale,Compromised->Mark()))*(1 - isRSSperfect) + CH2OKRate));
Timed Activity:	OK2FH
Distribution Parameters	Rate return(HazardRate * falseHazardProb);

Activation Predicate	(none)	
Reactivation Predicate	(none)	
Timed Activity:	OK2Hazard	
Distribution Parameters	Rate return(HazardRate * (1.0 - falseHazardProb));	
Activation Predicate	(none)	
Reactivation Predicate	(none)	
Case Distributions	case 1 return(1.0 - missHazardProb * pow(HazardScale,Compromised->Mark())); case 2 return(missHazardProb * pow(HazardScale,Compromised->Mark()));	
Timed Activity:	sojournTime	
Distribution Parameters	Rate return(OH2OKRate + MH2CHRate * pow(OH2CH_scale,Compromised->Mark()) + MH2AccRate * pow(DelayScale,Compromised->Mark()));	
Activation Predicate	(none)	
Reactivation Predicate	(none)	
Case Distributions	case 1 return((MH2AccRate * pow(DelayScale,Compromised->Mark()))/(OH2OKRate + MH2CHRate * pow(OH2CH_scale,Compromised->Mark()) + MH2AccRate * pow(DelayScale,Compromised->Mark()))); case 2 return((MH2CHRate * pow(OH2CH_scale,Compromised->Mark()))/(OH2OKRate + MH2CHRate * pow(OH2CH_scale,Compromised->Mark()) + MH2AccRate * pow(DelayScale,Compromised->Mark()))); case 3 return((OH2OKRate)/(OH2OKRate + MH2CHRate * pow(OH2CH_scale,Compromised->Mark()) + MH2AccRate * pow(DelayScale,Compromised->Mark())));	
Performance Variable Model: TimeToAccident		
Top Level Model Information	Child Model Name	Hazard_Attack
	Model Type	Rep/Join
Performance Variable : TimeToAccident		
Affecting Models	StateMachine	
Impulse Functions		
Reward Function	(Reward is over all Available Models) if(StateMachine->Accident->Mark() > 0) return(1); else return(0);	
Simulator Statistics	Type	Instant of Time
	Options	Estimate Mean
		Estimate Variance
		Include Lower Bound on Interval Estimate

		Include Upper Bound on Interval Estimate	
		Estimate Distribution	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	100.0,1100.0,2100.0,3100.0,4100.0,5100.0,6100.0,7100.0,8100.0,9100.0,
	Distributions	Lower Bound on Distribution Estimate	0.0
		Upper Bound on Distribution Estimate	1.0
		Step Size on Distribution Estimate	0.1
	Confidence	Confidence Level	0.95
		Confidence Interval	0.05

Set Study: set_st :			
Experiment	Variable	Type	Value
Experiment 1	AccidentScale	double	1.0
	CH2AccRate	double	1.0E-4
	CH2OKRate	double	856.3
	DelayScale	double	1.0
	FH2AccRate	double	1.0E-5
	FH2OKRate	double	856.3
	HazardRate	double	197.4
	HazardScale	double	1.0
	LateCH2AccScale	double	1.0
	MH2AccRate	double	0.001
	MH2CHRate	double	2250.0
	OH2CH_scale	double	1.0
	OH2OKRate	double	1000.0
	attackRate	double	0.1
	attackReset	double	2.0
	attackSuccess	double	0.2
	falseHazardProb	double	0.0
	isRSSperfect	int	0
	missHazardProb	double	0.05
Experiment 2	AccidentScale	double	10.0
	CH2AccRate	double	1.0E-4
	CH2OKRate	double	856.3
	DelayScale	double	1.0
	FH2AccRate	double	1.0E-5
	FH2OKRate	double	856.3
	HazardRate	double	197.4
	HazardScale	double	1.0

	LateCH2AccScale	double	1.0
	MH2AccRate	double	0.001
	MH2CHRate	double	2250.0
	OH2CH_scale	double	1.0
	OH2OKRate	double	1000.0
	attackRate	double	0.1
	attackReset	double	2.0
	attackSuccess	double	0.2
	falseHazardProb	double	0.0
	isRSSperfect	int	0
	missHazardProb	double	0.05
Experiment 3	AccidentScale	double	1.0
	CH2AccRate	double	1.0E-4
	CH2OKRate	double	856.3
	DelayScale	double	1000.0
	FH2AccRate	double	1.0E-5
	FH2OKRate	double	856.3
	HazardRate	double	197.4
	HazardScale	double	1.0
	LateCH2AccScale	double	1.0
	MH2AccRate	double	0.001
	MH2CHRate	double	2250.0
	OH2CH_scale	double	1.0
	OH2OKRate	double	1000.0
	attackRate	double	0.1
	attackReset	double	2.0
	attackSuccess	double	0.2
	falseHazardProb	double	0.0
	isRSSperfect	int	0
	missHazardProb	double	0.05
Experiment 4	AccidentScale	double	1.0
	CH2AccRate	double	1.0E-4
	CH2OKRate	double	856.3
	DelayScale	double	1.0
	FH2AccRate	double	1.0E-5
	FH2OKRate	double	856.3
	HazardRate	double	197.4
	HazardScale	double	10.0
	LateCH2AccScale	double	1.0
	MH2AccRate	double	0.001
	MH2CHRate	double	2250.0
	OH2CH_scale	double	1.0
	OH2OKRate	double	1000.0
	attackRate	double	0.1
	attackReset	double	2.0
	attackSuccess	double	0.2
	falseHazardProb	double	0.0

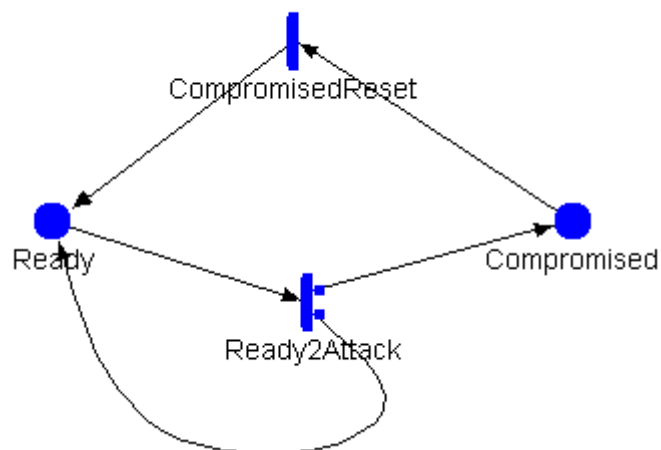
	isRSSperfect	int	0
	missHazardProb	double	0.05
Experiment 5	AccidentScale	double	1.0
	CH2AccRate	double	1.0E-4
	CH2OKRate	double	856.3
	DelayScale	double	1.0
	FH2AccRate	double	1.0E-5
	FH2OKRate	double	856.3
	HazardRate	double	197.4
	HazardScale	double	1.0
	LateCH2AccScale	double	10.0
	MH2AccRate	double	0.001
	MH2CHRate	double	2250.0
	OH2CH_scale	double	1.0
	OH2OKRate	double	1000.0
	attackRate	double	0.1
	attackReset	double	2.0
	attackSuccess	double	0.2
	falseHazardProb	double	0.0
	isRSSperfect	int	0
	missHazardProb	double	0.05
Experiment 6	AccidentScale	double	1.0
	CH2AccRate	double	1.0E-4
	CH2OKRate	double	856.3
	DelayScale	double	1.0
	FH2AccRate	double	1.0E-5
	FH2OKRate	double	856.3
	HazardRate	double	197.4
	HazardScale	double	1.0
	LateCH2AccScale	double	1.0
	MH2AccRate	double	0.001
	MH2CHRate	double	2250.0
	OH2CH_scale	double	10.0
	OH2OKRate	double	1000.0
	attackRate	double	0.1
	attackReset	double	2.0
	attackSuccess	double	0.2
	falseHazardProb	double	0.0
	isRSSperfect	int	0
	missHazardProb	double	0.05
Experiment 7	AccidentScale	double	10.0
	CH2AccRate	double	1.0E-4
	CH2OKRate	double	856.3
	DelayScale	double	1000.0
	FH2AccRate	double	1.0E-5
	FH2OKRate	double	856.3
	HazardRate	double	197.4

	HazardScale	double	10.0
	LateCH2AccScale	double	10.0
	MH2AccRate	double	0.001
	MH2CHRate	double	2250.0
	OH2CH_scale	double	10.0
	OH2OKRate	double	1000.0
	attackRate	double	0.1
	attackReset	double	2.0
	attackSuccess	double	0.2
	falseHazardProb	double	0.0
	isRSSperfect	int	0
	missHazardProb	double	0.05

HTML generated by Möbius using the “Document” feature.

13.2 Combined Vulnerability Modes Model

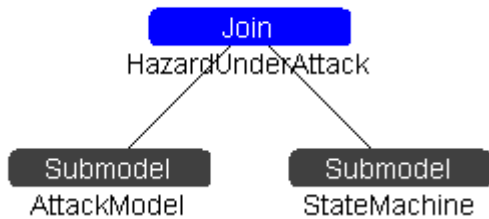
Model: AttackModel (Atomic Model)



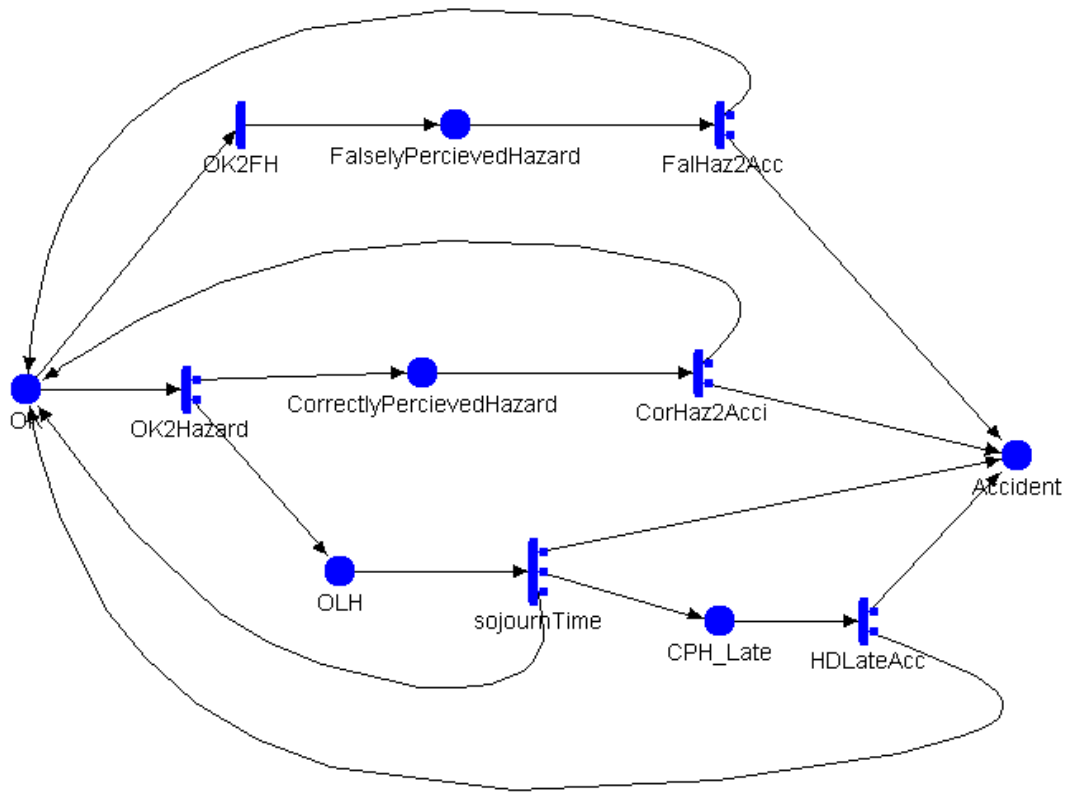
Place Attributes:	
Place Names	Initial Markings
Compromised	0
Ready	1
Timed Activity:	CompromisedReset
Distribution Parameters	Rate return(attackReset);
Activation Predicate	(none)
Reactivation Predicate	(none)
Timed Activity:	Ready2Attack
Distribution Parameters	Rate return(attackRate);
Activation Predicate	(none)
Reactivation Predicate	(none)

Case Distributions	case 1
	return(attackSuccess);
	case 2
	return(1.0 - attackSuccess);

Model: Hazard_Attack (Composed Model)



Model: StateMachine (Atomic Model)



isRSSperfect - a boolean variable:
 1: if RSS is perfect (detected hazard cannot lead to accident)
 0: if RSS is not perfect, then the hazard can lead to a an accident.

 Compromised

Place Attributes:	
Place Names	Initial Markings
Accident	0
CPH_Late	0

Compromised	0
CorrectlyPercievedHazard	0
FalselyPercievedHazard	0
OK	1
OLH	0
Timed Activity:	CorHaz2Acci
Distribution Parameters	Rate $\text{return}((\text{CH2AccRate} * \text{pow}(\text{AccidentScale}, \text{Compromised} \rightarrow \text{Mark}())) * (1 - \text{isRSSperfect}) + \text{CH2OKRate});$
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 $\text{return}(\text{CH2OKRate} / ((\text{CH2AccRate} * \text{pow}(\text{AccidentScale}, \text{Compromised} \rightarrow \text{Mark}())) * (1 - \text{isRSSperfect}) + \text{CH2OKRate}));$ case 2 $\text{return}((\text{CH2AccRate} * \text{pow}(\text{AccidentScale}, \text{Compromised} \rightarrow \text{Mark}())) * (1 - \text{isRSSperfect}) / ((\text{CH2AccRate} * \text{pow}(\text{AccidentScale}, \text{Compromised} \rightarrow \text{Mark}())) * (1 - \text{isRSSperfect}) + \text{CH2OKRate}));$
Timed Activity:	FalHaz2Acc
Distribution Parameters	Rate $\text{return}(\text{FH2AccRate} + \text{FH2OKRate});$
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 $\text{return}(\text{FH2OKRate} / (\text{FH2AccRate} * \text{pow}(\text{AccidentScale}, \text{Compromised} \rightarrow \text{Mark}()) + \text{FH2OKRate}));$ case 2 $\text{return}(\text{FH2AccRate} * \text{pow}(\text{AccidentScale}, \text{Compromised} \rightarrow \text{Mark}()) / (\text{FH2AccRate} * \text{pow}(\text{AccidentScale}, \text{Compromised} \rightarrow \text{Mark}()) + \text{FH2OKRate}));$
Timed Activity:	HDLateAcc
Distribution Parameters	Rate $\text{return}((\text{CH2AccRate} * \text{pow}(\text{LateCH2AccScale}, \text{Compromised} \rightarrow \text{Mark}())) * (1 - \text{isRSSperfect}) + \text{CH2OKRate});$
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 $\text{return}(((\text{CH2AccRate} * \text{pow}(\text{LateCH2AccScale}, \text{Compromised} \rightarrow \text{Mark}())) * (1 - \text{isRSSperfect})) / ((\text{CH2AccRate} * \text{pow}(\text{LateCH2AccScale}, \text{Compromised} \rightarrow \text{Mark}())) * (1 - \text{isRSSperfect}) + \text{CH2OKRate}));$ case 2

	return(CH2OKRate/((CH2AccRate * pow(LateCH2AccScale,Compromised->Mark()))*(1 - isRSSperfect) + CH2OKRate));	
Timed Activity:	OK2FH	
Distribution Parameters	Rate return(HazardRate * falseHazardProb);	
Activation Predicate	(none)	
Reactivation Predicate	(none)	
Timed Activity:	OK2Hazard	
Distribution Parameters	Rate return(HazardRate * (1.0 - falseHazardProb));	
Activation Predicate	(none)	
Reactivation Predicate	(none)	
Case Distributions	case 1 return(1.0 - missHazardProb * pow(HazardScale,Compromised->Mark())); case 2 return(missHazardProb * pow(HazardScale,Compromised->Mark()));	
Timed Activity:	sojournTime	
Distribution Parameters	Rate return(OH2OKRate + MH2CHRate * pow(OH2CH_scale,Compromised->Mark()) + MH2AccRate * pow(DelayScale,Compromised->Mark()));	
Activation Predicate	(none)	
Reactivation Predicate	(none)	
Case Distributions	case 1 return((MH2AccRate * pow(DelayScale,Compromised->Mark()))/(OH2OKRate + MH2CHRate * pow(OH2CH_scale,Compromised->Mark()) + MH2AccRate * pow(DelayScale,Compromised->Mark()))); case 2 return((MH2CHRate * pow(OH2CH_scale,Compromised->Mark()))/(OH2OKRate + MH2CHRate * pow(OH2CH_scale,Compromised->Mark()) + MH2AccRate * pow(DelayScale,Compromised->Mark()))); case 3 return((OH2OKRate)/(OH2OKRate + MH2CHRate * pow(OH2CH_scale,Compromised->Mark()) + MH2AccRate * pow(DelayScale,Compromised->Mark())));	
Performance Variable Model: TimeToAccident		
Top Level Model Information	Child Model Name	Hazard_Attack
	Model Type	Rep/Join
Performance Variable : TimeToAccident		
Affecting Models	StateMachine	
Impulse Functions		

Reward Function	(Reward is over all Available Models) if(StateMachine->Accident->Mark() > 0) return(1); else return(0);		
Simulator Statistics	Type	Instant of Time	
	Options	Estimate Mean	
		Estimate Variance	
		Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate Distribution	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	100.0,1100.0,2100.0,3100.0,4100.0,5100.0,6100.0,7100.0,8100.0,9100.0,
	Distributions	Lower Bound on Distribution Estimate	0.0
		Upper Bound on Distribution Estimate	1.0
		Step Size on Distribution Estimate	0.1
	Confidence	Confidence Level	0.95
		Confidence Interval	0.05

Range Study Variable Assignments for Study *range_final* in Project *CAV_Exp_v1* :

Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Manual	[1.0, 10.0]	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Manual	[1.0, 1000.0]	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Manual	[1.0, 10.0]	-	-	-	-
LateCH2AccScale	double	Manual	[1.0, 10.0]	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Manual	[1.0, 10.0]	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.1	-	-	-	-
attackReset	double	Fixed	2.0	-	-	-	-
attackSuccess	double	Fixed	0.2	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-

Range Study Variable Assignments for Study <i>range_real</i> in Project <i>CAV_Exp_v1</i> :							
Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Manual	[1.0, 10.0]	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Manual	[1.0, 1000.0]	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Manual	[1.0, 10.0]	-	-	-	-
LateCH2AccScale	double	Manual	[1.0, 10.0]	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Manual	[1.0, 10.0]	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.51	-	-	-	-
attackReset	double	Fixed	2.0	-	-	-	-
attackSuccess	double	Fixed	0.0031	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-
Range Study Variable Assignments for Study <i>range_sens_attrate</i> in Project <i>CAV_Exp_v1</i> :							
Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Manual	[1.0, 10.0]	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Manual	[1.0, 1000.0]	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Manual	[1.0, 10.0]	-	-	-	-
LateCH2AccScale	double	Manual	[1.0, 10.0]	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Manual	[1.0, 10.0]	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Manual	[0.1, 0.2, 0.3]	-	-	-	-
attackReset	double	Fixed	1.0E-4	-	-	-	-
attackSuccess	double	Fixed	0.5	-	-	-	-
falseHazardProb	double	Fixed	0.0	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-
Range Study Variable Assignments for Study <i>range_sens_attreset</i> in Project <i>CAV_Exp_v1</i> :							
Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Manual	[1.0, 10.0]	-	-	-	-

CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Manual	[1.0, 1000.0]	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Manual	[1.0, 10.0]	-	-	-	-
LateCH2AccScale	double	Manual	[1.0, 10.0]	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Manual	[1.0, 10.0]	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.1	-	-	-	-
attackReset	double	Manual	[0.01, 1.0, 2.0]	-	-	-	-
attackSuccess	double	Fixed	0.5	-	-	-	-
falseHazardProb	double	Fixed	0.0	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-

Range Study Variable Assignments for Study *range_sens_attsuccess* in Project *CAV_Exp_v1* :

Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Manual	[1.0, 10.0]	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Manual	[1.0, 1000.0]	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Manual	[1.0, 10.0]	-	-	-	-
LateCH2AccScale	double	Manual	[1.0, 10.0]	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Manual	[1.0, 10.0]	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.1	-	-	-	-
attackReset	double	Fixed	1.0E-4	-	-	-	-
attackSuccess	double	Manual	[0.05, 0.1, 0.2]	-	-	-	-
falseHazardProb	double	Fixed	0.0	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-

Range Study Variable Assignments for Study *range_sens_fh* in Project *CAV_Exp_v1* :

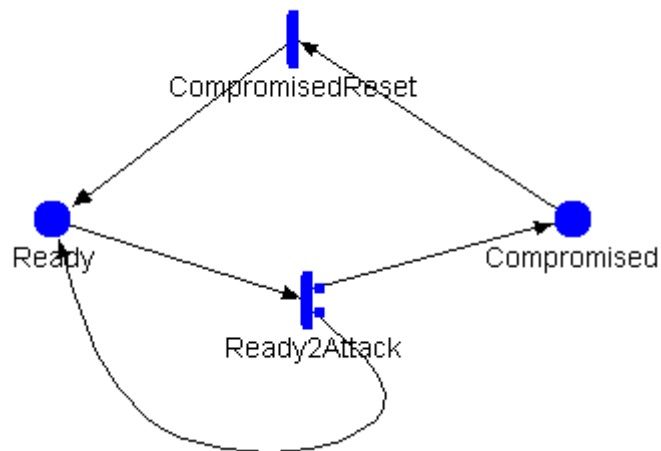
Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Manual	[1.0, 10.0]	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Manual	[1.0, 1000.0]	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-

FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Manual	[1.0, 10.0]	-	-	-	-
LateCH2AccScale	double	Manual	[1.0, 10.0]	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Manual	[1.0, 10.0]	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.1	-	-	-	-
attackReset	double	Fixed	1.0E-4	-	-	-	-
attackSuccess	double	Fixed	0.5	-	-	-	-
falseHazardProb	double	Manual	[0.0, 1.0E-4, 0.001]	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-

HTML generated by Möbius using the “Document” feature.

13.3 Partitioned Vehicle Architecture Model

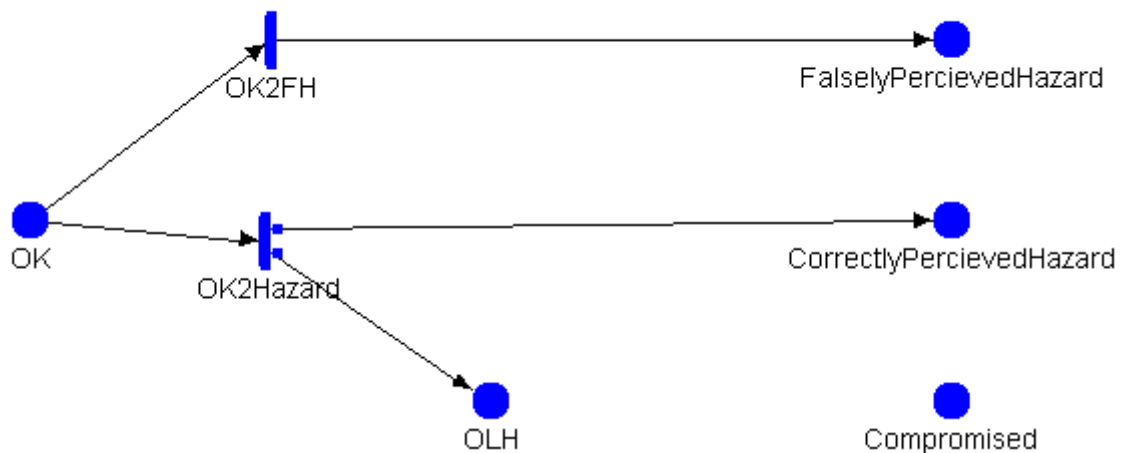
Model: AttackModel (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
Compromised	0
Ready	1
Timed Activity:	CompromisedReset
Distribution Parameters	Rate return(attackReset);
Activation Predicate	(none)
Reactivation Predicate	(none)
Timed Activity:	Ready2Attack

Distribution Parameters	Rate return(attackRate);
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 return(attackSuccess); case 2 return(1.0 - attackSuccess);

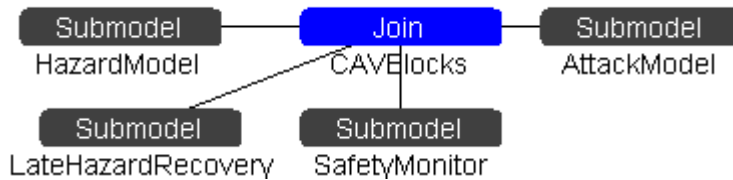
Model: HazardModel (Atomic Model)



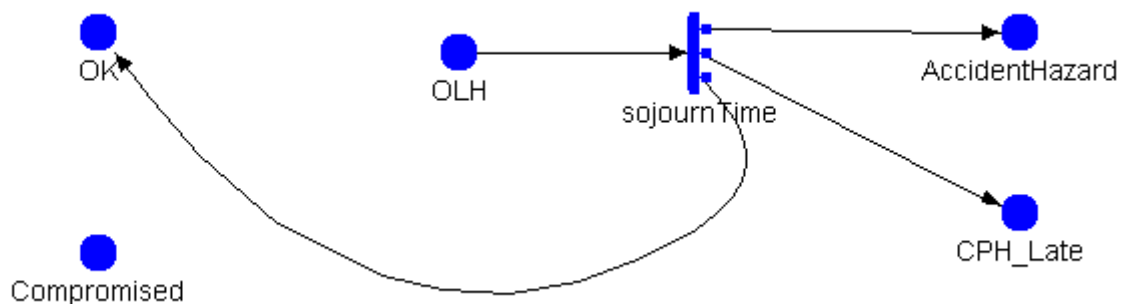
Place Attributes:	
Place Names	Initial Markings
Compromised	0
CorrectlyPercievedHazard	0
FalselyPercievedHazard	0
OK	1
OLH	0
Timed Activity:	OK2FH
Distribution Parameters	Rate return(HazardRate * falseHazardProb);
Activation Predicate	(none)
Reactivation Predicate	(none)
Timed Activity:	OK2Hazard
Distribution Parameters	Rate return(HazardRate * (1.0 - falseHazardProb));
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 return(1.0 - missHazardProb * pow(HazardScale,Compromised->Mark() && enableHazard)); case 2

	<pre>return(missHazardProb * pow(HazardScale,Compromised->Mark() && enableHazard));</pre>
--	--

Model: HazardSystemBlocks (Composed Model)



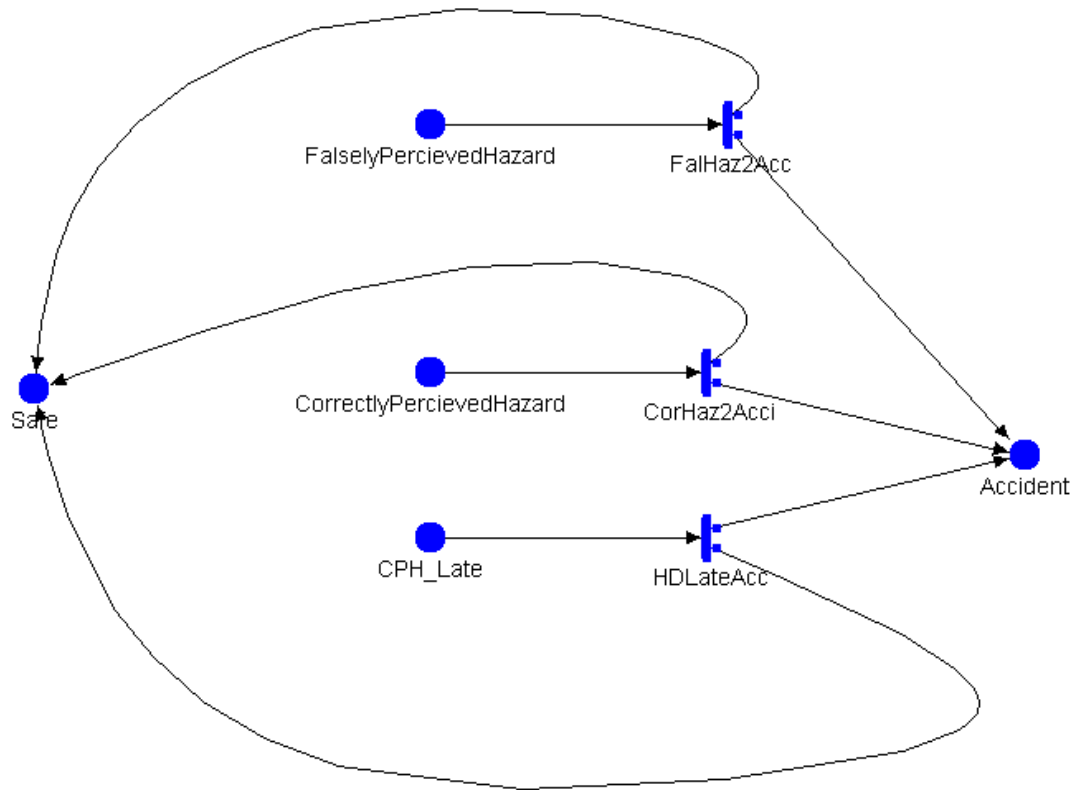
Model: LateHazardRecovery (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
AccidentHazard	0
CPH_Late	0
Compromised	0
OK	1
OLH	0
Timed Activity:	sojournTime
Distribution Parameters	Rate <pre>return(OH2OKRate + MH2CHRate * pow(OH2CH_scale,Compromised->Mark() && enableOLH) + MH2AccRate * pow(DelayScale,Compromised->Mark() && enableOLH));</pre>
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 <pre>return((MH2AccRate * pow(DelayScale,Compromised->Mark() && enableOLH))/(OH2OKRate + MH2CHRate * pow(OH2CH_scale,Compromised->Mark() && enableOLH) + MH2AccRate * pow(DelayScale,Compromised->Mark() && enableOLH));</pre> case 2 <pre>return((MH2CHRate * pow(OH2CH_scale,Compromised->Mark() && enableOLH))/(OH2OKRate + MH2CHRate * pow(OH2CH_scale,Compromised->Mark() && enableOLH) + MH2AccRate * pow(DelayScale,Compromised->Mark() && enableOLH));</pre>

	case 3 $\text{return}((\text{OH2OKRate})/(\text{OH2OKRate} + \text{MH2CHRate} * \text{pow}(\text{OH2CH_scale}, \text{Compromised} \rightarrow \text{Mark}() \ \&\& \ \text{enableOLH}) + \text{MH2AccRate} * \text{pow}(\text{DelayScale}, \text{Compromised} \rightarrow \text{Mark}() \ \&\& \ \text{enableOLH})));$
--	---

Model: SafetyMonitor (Atomic Model)



isRSSperfect - a boolean variable:

1: if RSS is perfect (detected hazard cannot lead to accident)

0: if RSS is not perfect, then the hazard can lead to a an accident.

 Compromised

Place Attributes:	
Place Names	Initial Markings
Accident	0
CPH_Late	0
Compromised	0
CorrectlyPercievedHazard	0
FalselyPercievedHazard	0
Safe	1
Timed Activity:	CorHaz2Acci
Distribution Parameters	Rate $\text{return}((\text{CH2AccRate} * \text{pow}(\text{AccidentScale}, \text{Compromised} \rightarrow \text{Mark}() \ \&\& \ \text{enableSM})) * (1 - \text{isRSSperfect}) + \text{CH2OKRate});$

Activation Predicate	(none)	
Reactivation Predicate	(none)	
Case Distributions	case 1 return(CH2OKRate/((CH2AccRate * pow(AccidentScale,Compromised->Mark() && enableSM))*(1 - isRSSperfect) + CH2OKRate)); case 2 return(((CH2AccRate * pow(AccidentScale,Compromised->Mark() && enableSM))*(1 - isRSSperfect)/((CH2AccRate * pow(AccidentScale,Compromised->Mark() && enableSM))*(1 - isRSSperfect) + CH2OKRate));	
Timed Activity:	FalHaz2Acc	
Distribution Parameters	Rate return(FH2AccRate + FH2OKRate);	
Activation Predicate	(none)	
Reactivation Predicate	(none)	
Case Distributions	case 1 return(FH2OKRate / (FH2AccRate * pow(AccidentScale,Compromised->Mark() && enableSM) + FH2OKRate)); case 2 return(FH2AccRate * pow(AccidentScale,Compromised->Mark() && enableSM) / (FH2AccRate * pow(AccidentScale,Compromised->Mark() && enableSM) + FH2OKRate));	
Timed Activity:	HDLateAcc	
Distribution Parameters	Rate return(((CH2AccRate * pow(LateCH2AccScale,Compromised->Mark() && enableSM))*(1 - isRSSperfect) + CH2OKRate);	
Activation Predicate	(none)	
Reactivation Predicate	(none)	
Case Distributions	case 1 return((((CH2AccRate * pow(LateCH2AccScale,Compromised->Mark() && enableSM))*(1 - isRSSperfect)/((CH2AccRate * pow(LateCH2AccScale,Compromised->Mark() && enableSM))*(1 - isRSSperfect) + CH2OKRate)); case 2 return(CH2OKRate/((CH2AccRate * pow(LateCH2AccScale,Compromised->Mark() && enableSM))*(1 - isRSSperfect) + CH2OKRate));	
Performance Variable Model: TimeToAccident		
Top Level Model Information	Child Model Name	HazardSystemBlocks
	Model Type	Rep/Join
Performance Variable : TimeToAccident		
Affecting Models	SafetyMonitor	

Impulse Functions			
Reward Function	<i>(Reward is over all Available Models)</i> if(SafetyMonitor->Accident->Mark() > 0) return(1); else return(0);		
Simulator Statistics	Type	Instant of Time	
	Options	Estimate Mean	
		Estimate Variance	
		Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate Distribution	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	100.0,1100.0,2100.0,3100.0,4100.0,5100.0,6100.0,7100.0,8100.0,9100.0,
	Distributions	Lower Bound on Distribution Estimate	0.0
		Upper Bound on Distribution Estimate	1.0
		Step Size on Distribution Estimate	0.1
	Confidence	Confidence Level	0.95
		Confidence Interval	0.05

Range Study Variable Assignments for Study *range_st* in Project *CAV_Exp_v2* :

Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.1	-	-	-	-
attackReset	double	Fixed	2.0	-	-	-	-
attackSuccess	double	Fixed	0.2	-	-	-	-
enableHazard	short	Manual	[0, 1]	-	-	-	-

enableOLH	short	Manual	[0, 1]	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-

Range Study Variable Assignments for Study *range_st_sensreset* in Project *CAV_Exp_v2* :

Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.1	-	-	-	-
attackReset	double	Manual	[0.001, 0.01, 0.1, 1.0, 2.0, 4.0]	-	-	-	-
attackSuccess	double	Fixed	0.2	-	-	-	-
enableHazard	short	Manual	[0, 1]	-	-	-	-
enableOLH	short	Manual	[0, 1]	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-

Range Study Variable Assignments for Study *range_streal* in Project *CAV_Exp_v2* :

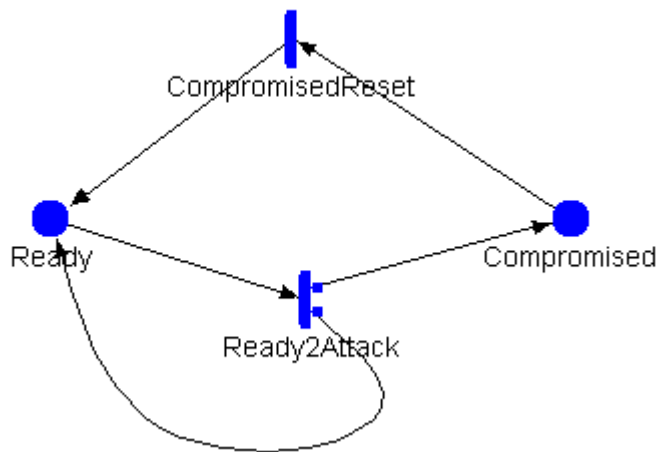
Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.51	-	-	-	-

attackReset	double	Fixed	2.0	-	-	-	-
attackSuccess	double	Fixed	0.0031	-	-	-	-
enableHazard	short	Manual	[0, 1]	-	-	-	-
enableOLH	short	Manual	[0, 1]	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-

HTML generated by Möbius using the “Document” feature.

13.4 Defence-in-Depth Model

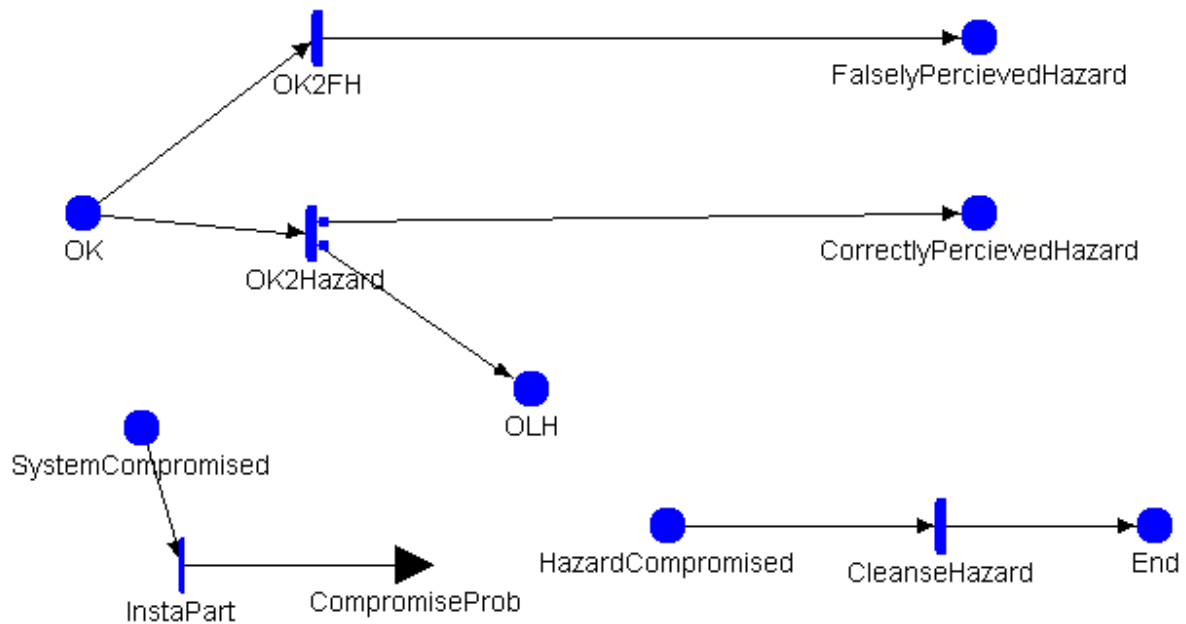
Model: AttackModel (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
Compromised	0
Ready	1
Timed Activity:	CompromisedReset
Distribution Parameters	Rate return(attackReset);
Activation Predicate	(none)
Reactivation Predicate	(none)
Timed Activity:	Ready2Attack
Distribution Parameters	Rate return(attackRate);
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 return(attackSuccess); case 2

	return(1.0 - attackSuccess);
--	------------------------------

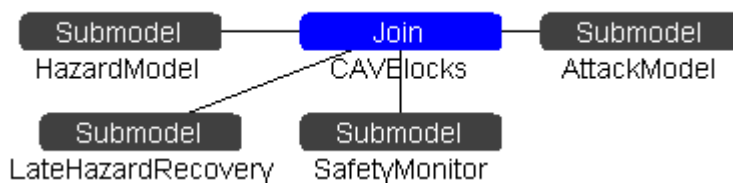
Model: HazardModel (Atomic Model)



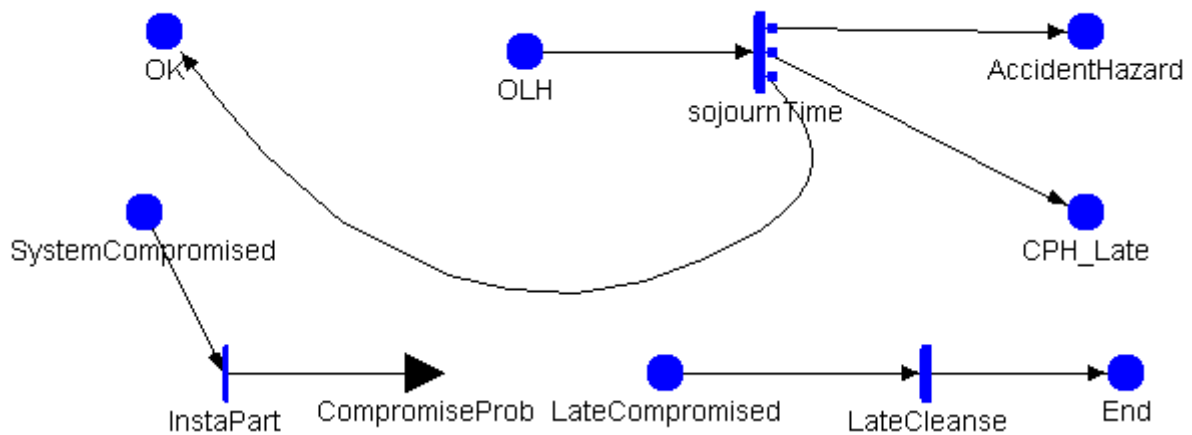
Place Attributes:	
Place Names	Initial Markings
CorrectlyPercievedHazard	0
End	0
FalselyPercievedHazard	0
HazardCompromised	0
OK	1
OLH	0
SystemCompromised	0
Timed Activity:	CleanseHazard
Distribution Parameters	Rate return(hazCleanseRate);
Activation Predicate	(none)
Reactivation Predicate	(none)
Timed Activity:	OK2FH
Distribution Parameters	Rate return(HazardRate * falseHazardProb);
Activation Predicate	(none)
Reactivation Predicate	(none)
Timed Activity:	OK2Hazard
Distribution Parameters	Rate return(HazardRate * (1.0 - falseHazardProb));
Activation Predicate	(none)
Reactivation Predicate	(none)

Case Distributions	case 1 return(1.0 - missHazardProb * pow(HazardScale,HazardCompromised->Mark() && enableHazard));
	case 2 return(missHazardProb * pow(HazardScale,HazardCompromised->Mark() && enableHazard));
Instantaneous Activities Without Cases:	
InstaPart	
Output Gate:	CompromiseProb
Function	static UserDistributions dist(0,31415); double p = dist.Uniform(0,1); if (p <= hazSuccessProb) HazardCompromised->Mark()=1;

Model: HazardSystemBlocks (Composed Model)



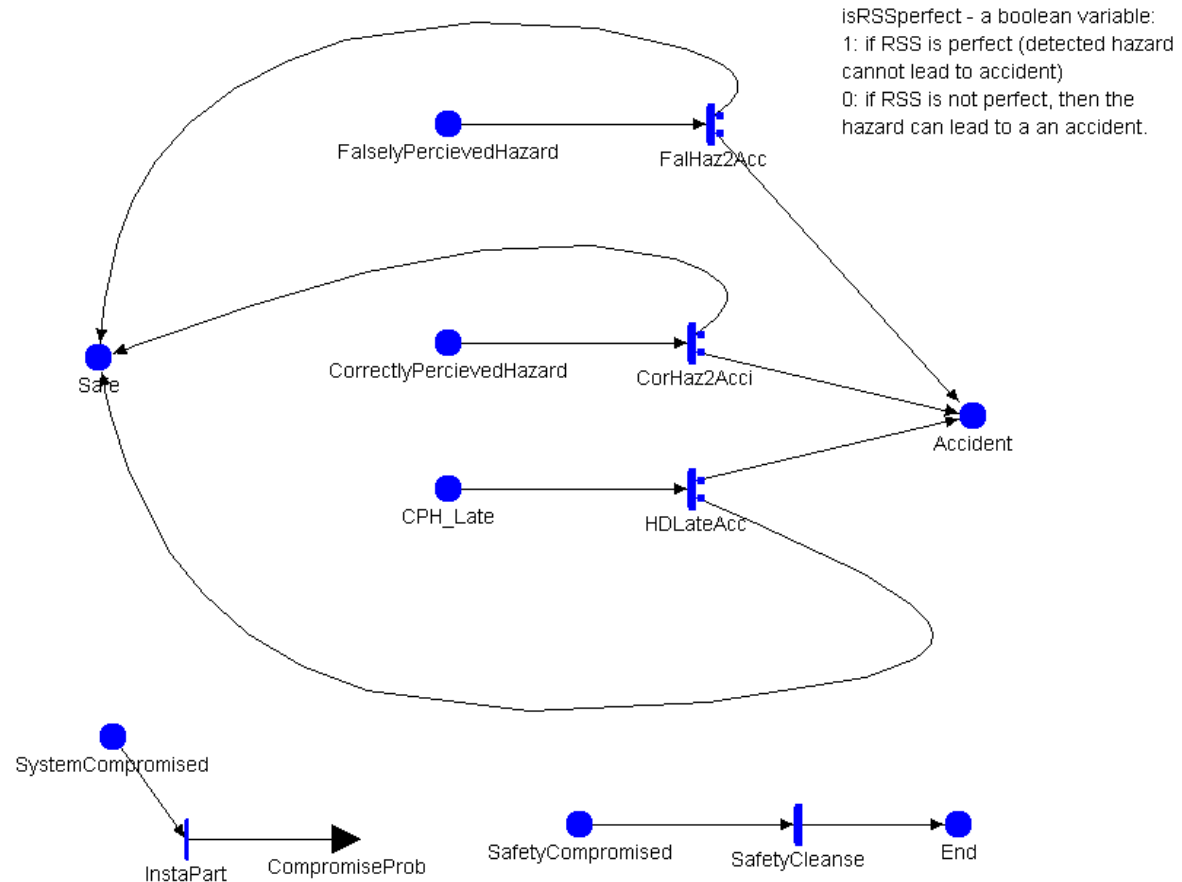
Model: LateHazardRecovery (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
AccidentHazard	0
CPH_Late	0
End	0
LateCompromised	0
OK	1
OLH	0
SystemCompromised	0
Timed Activity:	LateCleanse

Distribution Parameters	Rate return(olhCleanseRate);
Activation Predicate	(none)
Reactivation Predicate	(none)
Timed Activity:	sojournTime
Distribution Parameters	Rate return(OH2OKRate + MH2CHRate * pow(OH2CH_scale,LateCompromised->Mark() && enableOLH) + MH2AccRate * pow(DelayScale,LateCompromised->Mark() && enableOLH));
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 return((MH2AccRate * pow(DelayScale,LateCompromised->Mark() && enableOLH))/(OH2OKRate + MH2CHRate * pow(OH2CH_scale,LateCompromised->Mark() && enableOLH) + MH2AccRate * pow(DelayScale,LateCompromised->Mark() && enableOLH))); case 2 return((MH2CHRate * pow(OH2CH_scale,LateCompromised->Mark() && enableOLH))/(OH2OKRate + MH2CHRate * pow(OH2CH_scale,LateCompromised->Mark() && enableOLH) + MH2AccRate * pow(DelayScale,LateCompromised->Mark() && enableOLH))); case 3 return(((OH2OKRate)/(OH2OKRate + MH2CHRate * pow(OH2CH_scale,LateCompromised->Mark() && enableOLH) + MH2AccRate * pow(DelayScale,LateCompromised->Mark() && enableOLH)));
Instantaneous Activities Without Cases:	
InstaPart	
Output Gate:	CompromiseProb
Function	static UserDistributions dist(0,31415); double p = dist.Uniform(0,1); if (p <= olhSuccessProb) LateCompromised->Mark()=1;

Model: SafetyMonitor (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
Accident	0
CPH_Late	0
CorrectlyPercievedHazard	0
End	0
FalselyPercievedHazard	0
Safe	1
SafetyCompromised	0
SystemCompromised	0
Timed Activity:	CorHaz2Acci
Distribution Parameters	Rate $\text{return}((\text{CH2AccRate} * \text{pow}(\text{AccidentScale}, \text{SafetyCompromised} \rightarrow \text{Mark}()) \&\& \text{enableSM})) * (1 - \text{isRSSperfect}) + \text{CH2OKRate});$
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 $\text{return}(\text{CH2OKRate} / ((\text{CH2AccRate} * \text{pow}(\text{AccidentScale}, \text{SafetyCompromised} \rightarrow \text{Mark}()) \&\& \text{enableSM})) * (1 - \text{isRSSperfect}) + \text{CH2OKRate});$

	case 2 return((CH2AccRate * pow(AccidentScale,SafetyCompromised->Mark() && enableSM))*(1 - isRSSperfect)/((CH2AccRate * pow(AccidentScale,SafetyCompromised->Mark() && enableSM))*(1 - isRSSperfect) + CH2OKRate));
Timed Activity:	FalHaz2Acc
Distribution Parameters	Rate return(FH2AccRate + FH2OKRate);
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 return(FH2OKRate / (FH2AccRate * pow(AccidentScale,SafetyCompromised->Mark() && enableSM) + FH2OKRate)); case 2 return(FH2AccRate * pow(AccidentScale,SafetyCompromised->Mark() && enableSM) / (FH2AccRate * pow(AccidentScale,SafetyCompromised->Mark() && enableSM) + FH2OKRate));
Timed Activity:	HDLateAcc
Distribution Parameters	Rate return((CH2AccRate * pow(LateCH2AccScale,SafetyCompromised->Mark() && enableSM))*(1 - isRSSperfect) + CH2OKRate);
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 return(((CH2AccRate * pow(LateCH2AccScale,SafetyCompromised->Mark() && enableSM))*(1 - isRSSperfect))/((CH2AccRate * pow(LateCH2AccScale,SafetyCompromised->Mark() && enableSM))*(1 - isRSSperfect) + CH2OKRate)); case 2 return(CH2OKRate/((CH2AccRate * pow(LateCH2AccScale,SafetyCompromised->Mark() && enableSM))*(1 - isRSSperfect) + CH2OKRate));
Timed Activity:	SafetyCleanse
Distribution Parameters	Rate return(safeCleanseRate);
Activation Predicate	(none)
Reactivation Predicate	(none)
Instantaneous Activities Without Cases:	
InstaPart	
Output Gate:	CompromiseProb
Function	static UserDistributions dist(0,31415); double p = dist.Uniform(0,1); if (p <= safeSuccessProb) SafetyCompromised->Mark()=1;

Performance Variable Model: TimeToAccident							
Top Level Model Information		Child Model Name	HazardSystemBlocks				
		Model Type	Rep/Join				
Performance Variable : TimeToAccident							
Affecting Models	SafetyMonitor						
Impulse Functions							
Reward Function	(Reward is over all Available Models) if(SafetyMonitor->Accident->Mark() > 0) return(1); else return(0);						
Simulator Statistics	Type	Instant of Time					
	Options	Estimate Mean					
		Estimate Variance					
		Include Lower Bound on Interval Estimate					
		Include Upper Bound on Interval Estimate					
		Estimate Distribution					
		Estimate out of Range Probabilities					
		Confidence Level is Relative					
	Parameters	Start Time	100.0,1100.0,2100.0,3100.0,4100.0,5100.0,6100.0,7100.0,8100.0,9100.0,				
	Distributions	Lower Bound on Distribution Estimate	0.0				
		Upper Bound on Distribution Estimate	1.0				
		Step Size on Distribution Estimate	0.1				
	Confidence	Confidence Level	0.95				
		Confidence Interval	0.05				
Range Study Variable Assignments for Study <i>st_cprob_haz</i> in Project <i>CAV_Exp_v3</i> :							
Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-

OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.1	-	-	-	-
attackReset	double	Fixed	2.0	-	-	-	-
attackSuccess	double	Fixed	0.3	-	-	-	-
enableHazard	short	Manual	[0, 1]	-	-	-	-
enableOLH	short	Manual	[0, 1]	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
hazCleanseRate	double	Fixed	0.001	-	-	-	-
hazSuccessProb	double	Fixed	0.2	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-
olhCleanseRate	double	Fixed	0.001	-	-	-	-
olhSuccessProb	double	Fixed	0.1	-	-	-	-
safeCleanseRate	double	Fixed	0.001	-	-	-	-
safeSuccessProb	double	Fixed	0.5	-	-	-	-

Range Study Variable Assignments for Study *st_cprob_olh* in Project *CAV_Exp_v3* :

Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.1	-	-	-	-
attackReset	double	Fixed	2.0	-	-	-	-
attackSuccess	double	Fixed	0.1	-	-	-	-
enableHazard	short	Manual	[0, 1]	-	-	-	-
enableOLH	short	Manual	[0, 1]	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
hazCleanseRate	double	Fixed	0.001	-	-	-	-
hazSuccessProb	double	Fixed	0.3	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-
olhCleanseRate	double	Fixed	0.001	-	-	-	-
olhSuccessProb	double	Fixed	0.2	-	-	-	-
safeCleanseRate	double	Fixed	0.001	-	-	-	-
safeSuccessProb	double	Fixed	0.5	-	-	-	-

Range Study Variable Assignments for Study <i>st_cprob_real</i> in Project <i>CAV_Exp_v3</i> :							
Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.51	-	-	-	-
attackReset	double	Fixed	2.0	-	-	-	-
attackSuccess	double	Fixed	0.2209	-	-	-	-
enableHazard	short	Manual	[0, 1]	-	-	-	-
enableOLH	short	Manual	[0, 1]	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
hazCleanseRate	double	Fixed	0.001	-	-	-	-
hazSuccessProb	double	Fixed	0.0031	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-
olhCleanseRate	double	Fixed	0.001	-	-	-	-
olhSuccessProb	double	Fixed	0.0031	-	-	-	-
safeCleanseRate	double	Fixed	0.001	-	-	-	-
safeSuccessProb	double	Fixed	0.0031	-	-	-	-
Range Study Variable Assignments for Study <i>st_cprob_realhaz</i> in Project <i>CAV_Exp_v3</i> :							
Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.51	-	-	-	-

attackReset	double	Fixed	2.0	-	-	-	-
attackSuccess	double	Fixed	0.2209	-	-	-	-
enableHazard	short	Manual	[0, 1]	-	-	-	-
enableOLH	short	Manual	[0, 1]	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
hazCleanseRate	double	Fixed	0.001	-	-	-	-
hazSuccessProb	double	Fixed	0.0031	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-
olhCleanseRate	double	Fixed	0.001	-	-	-	-
olhSuccessProb	double	Fixed	0.2209	-	-	-	-
safeCleanseRate	double	Fixed	0.001	-	-	-	-
safeSuccessProb	double	Fixed	0.2209	-	-	-	-

Range Study Variable Assignments for Study *st_cprob_realolh* in Project *CAV_Exp_v3* :

Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.51	-	-	-	-
attackReset	double	Fixed	2.0	-	-	-	-
attackSuccess	double	Fixed	0.2209	-	-	-	-
enableHazard	short	Manual	[0, 1]	-	-	-	-
enableOLH	short	Manual	[0, 1]	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
hazCleanseRate	double	Fixed	0.001	-	-	-	-
hazSuccessProb	double	Fixed	0.2209	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-
olhCleanseRate	double	Fixed	0.001	-	-	-	-
olhSuccessProb	double	Fixed	0.0031	-	-	-	-
safeCleanseRate	double	Fixed	0.001	-	-	-	-
safeSuccessProb	double	Fixed	0.2209	-	-	-	-

Range Study Variable Assignments for Study *st_cprob_realsm* in Project *CAV_Exp_v3* :

Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
----------	------	------------	-------	-----------	----------------	----------	---

AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.51	-	-	-	-
attackReset	double	Fixed	2.0	-	-	-	-
attackSuccess	double	Fixed	0.2209	-	-	-	-
enableHazard	short	Manual	[0, 1]	-	-	-	-
enableOLH	short	Manual	[0, 1]	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
hazCleanseRate	double	Fixed	0.001	-	-	-	-
hazSuccessProb	double	Fixed	0.2209	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-
olhCleanseRate	double	Fixed	0.001	-	-	-	-
olhSuccessProb	double	Fixed	0.2209	-	-	-	-
safeCleanseRate	double	Fixed	0.001	-	-	-	-
safeSuccessProb	double	Fixed	0.0031	-	-	-	-

Range Study Variable Assignments for Study *st_cprob_sm* in Project *CAV_Exp_v3* :

Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.1	-	-	-	-
attackReset	double	Fixed	2.0	-	-	-	-
attackSuccess	double	Fixed	0.5	-	-	-	-
enableHazard	short	Manual	[0, 1]	-	-	-	-

enableOLH	short	Manual	[0, 1]	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
hazCleanseRate	double	Fixed	0.001	-	-	-	-
hazSuccessProb	double	Fixed	0.3	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-
olhCleanseRate	double	Fixed	0.001	-	-	-	-
olhSuccessProb	double	Fixed	0.1	-	-	-	-
safeCleanseRate	double	Fixed	0.001	-	-	-	-
safeSuccessProb	double	Fixed	0.2	-	-	-	-

Range Study Variable Assignments for Study *st_final* in Project *CAV_Exp_v3* :

Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.1	-	-	-	-
attackReset	double	Fixed	2.0	-	-	-	-
attackSuccess	double	Fixed	0.2	-	-	-	-
enableHazard	short	Manual	[0, 1]	-	-	-	-
enableOLH	short	Manual	[0, 1]	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
hazCleanseRate	double	Fixed	0.001	-	-	-	-
hazSuccessProb	double	Fixed	0.3	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-
olhCleanseRate	double	Fixed	0.001	-	-	-	-
olhSuccessProb	double	Fixed	0.1	-	-	-	-
safeCleanseRate	double	Fixed	0.001	-	-	-	-
safeSuccessProb	double	Fixed	0.5	-	-	-	-

Range Study Variable Assignments for Study *st_real* in Project *CAV_Exp_v3* :

Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-

DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.51	-	-	-	-
attackReset	double	Fixed	2.0	-	-	-	-
attackSuccess	double	Fixed	0.0031	-	-	-	-
enableHazard	short	Manual	[0, 1]	-	-	-	-
enableOLH	short	Manual	[0, 1]	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
hazCleanseRate	double	Fixed	0.001	-	-	-	-
hazSuccessProb	double	Fixed	0.2209	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-
olhCleanseRate	double	Fixed	0.001	-	-	-	-
olhSuccessProb	double	Fixed	0.2209	-	-	-	-
safeCleanseRate	double	Fixed	0.001	-	-	-	-
safeSuccessProb	double	Fixed	0.2209	-	-	-	-

Range Study Variable Assignments for Study *st_sens_attreset* in Project *CAV_Exp_v3* :

Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.1	-	-	-	-
attackReset	double	Manual	[0.001, 0.01, 0.1, 1.0, 2.0, 4.0]	-	-	-	-
attackSuccess	double	Fixed	0.2	-	-	-	-
enableHazard	short	Manual	[0, 1]	-	-	-	-
enableOLH	short	Manual	[0, 1]	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-

falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
hazCleanseRate	double	Fixed	0.001	-	-	-	-
hazSuccessProb	double	Fixed	1.0	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-
olhCleanseRate	double	Fixed	0.001	-	-	-	-
olhSuccessProb	double	Fixed	1.0	-	-	-	-
safeCleanseRate	double	Fixed	0.001	-	-	-	-
safeSuccessProb	double	Fixed	1.0	-	-	-	-

Range Study Variable Assignments for Study *st_sens_hazcleanse* in Project *CAV_Exp_v3* :

Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.1	-	-	-	-
attackReset	double	Fixed	2.0	-	-	-	-
attackSuccess	double	Fixed	0.2	-	-	-	-
enableHazard	short	Manual	[0, 1]	-	-	-	-
enableOLH	short	Manual	[0, 1]	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
hazCleanseRate	double	Manual	[0.001, 0.01, 0.1, 1.0, 2.0, 4.0]	-	-	-	-
hazSuccessProb	double	Fixed	1.0	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-
olhCleanseRate	double	Fixed	0.001	-	-	-	-
olhSuccessProb	double	Fixed	1.0	-	-	-	-
safeCleanseRate	double	Fixed	0.001	-	-	-	-
safeSuccessProb	double	Fixed	1.0	-	-	-	-

Range Study Variable Assignments for Study *st_sens_hazprob* in Project *CAV_Exp_v3* :

Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-

FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.1	-	-	-	-
attackReset	double	Fixed	2.0	-	-	-	-
attackSuccess	double	Fixed	0.2	-	-	-	-
enableHazard	short	Manual	[0, 1]	-	-	-	-
enableOLH	short	Manual	[0, 1]	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
hazCleanseRate	double	Fixed	0.001	-	-	-	-
hazSuccessProb	double	Manual	[0.1, 0.3, 0.5]	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-
olhCleanseRate	double	Fixed	0.001	-	-	-	-
olhSuccessProb	double	Fixed	1.0	-	-	-	-
safeCleanseRate	double	Fixed	0.001	-	-	-	-
safeSuccessProb	double	Fixed	1.0	-	-	-	-

Range Study Variable Assignments for Study *st_sens_olhcleanse* in Project *CAV_Exp_v3* :

Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.1	-	-	-	-
attackReset	double	Fixed	2.0	-	-	-	-
attackSuccess	double	Fixed	0.2	-	-	-	-
enableHazard	short	Manual	[0, 1]	-	-	-	-
enableOLH	short	Manual	[0, 1]	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
hazCleanseRate	double	Fixed	0.001	-	-	-	-

hazSuccessProb	double	Fixed	1.0	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-
olhCleanseRate	double	Manual	[0.001, 0.01, 0.1, 1.0, 2.0, 4.0]	-	-	-	-
olhSuccessProb	double	Fixed	1.0	-	-	-	-
safeCleanseRate	double	Fixed	0.001	-	-	-	-
safeSuccessProb	double	Fixed	1.0	-	-	-	-

Range Study Variable Assignments for Study *st_sens_olhprob* in Project *CAV_Exp_v3* :

Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.1	-	-	-	-
attackReset	double	Fixed	2.0	-	-	-	-
attackSuccess	double	Fixed	0.2	-	-	-	-
enableHazard	short	Manual	[0, 1]	-	-	-	-
enableOLH	short	Manual	[0, 1]	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
hazCleanseRate	double	Fixed	0.001	-	-	-	-
hazSuccessProb	double	Fixed	1.0	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-
olhCleanseRate	double	Fixed	0.001	-	-	-	-
olhSuccessProb	double	Manual	[0.1, 0.3, 0.5]	-	-	-	-
safeCleanseRate	double	Fixed	0.001	-	-	-	-
safeSuccessProb	double	Fixed	1.0	-	-	-	-

Range Study Variable Assignments for Study *st_sens_safecleanse* in Project *CAV_Exp_v3* :

Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-

HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.1	-	-	-	-
attackReset	double	Fixed	2.0	-	-	-	-
attackSuccess	double	Fixed	0.2	-	-	-	-
enableHazard	short	Manual	[0, 1]	-	-	-	-
enableOLH	short	Manual	[0, 1]	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
hazCleanseRate	double	Fixed	0.001	-	-	-	-
hazSuccessProb	double	Fixed	1.0	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-
olhCleanseRate	double	Fixed	0.001	-	-	-	-
olhSuccessProb	double	Fixed	1.0	-	-	-	-
safeCleanseRate	double	Manual	[0.001, 0.01, 0.1, 1.0, 2.0, 4.0]	-	-	-	-
safeSuccessProb	double	Fixed	1.0	-	-	-	-

Range Study Variable Assignments for Study *st_sens_safeprob* in Project *CAV_Exp_v3* :

Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.1	-	-	-	-
attackReset	double	Fixed	2.0	-	-	-	-
attackSuccess	double	Fixed	0.2	-	-	-	-
enableHazard	short	Manual	[0, 1]	-	-	-	-
enableOLH	short	Manual	[0, 1]	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
hazCleanseRate	double	Fixed	0.001	-	-	-	-
hazSuccessProb	double	Fixed	1.0	-	-	-	-

isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-
olhCleanseRate	double	Fixed	0.001	-	-	-	-
olhSuccessProb	double	Fixed	1.0	-	-	-	-
safeCleanseRate	double	Fixed	0.001	-	-	-	-
safeSuccessProb	double	Manual	[0.1, 0.3, 0.5]	-	-	-	-

Range Study Variable Assignments for Study *st_strongatt* in Project *CAV_Exp_v3* :

Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.3	-	-	-	-
attackReset	double	Fixed	2.0	-	-	-	-
attackSuccess	double	Fixed	0.75	-	-	-	-
enableHazard	short	Manual	[0, 1]	-	-	-	-
enableOLH	short	Manual	[0, 1]	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
hazCleanseRate	double	Fixed	0.001	-	-	-	-
hazSuccessProb	double	Fixed	0.3	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-
olhCleanseRate	double	Fixed	0.001	-	-	-	-
olhSuccessProb	double	Fixed	0.1	-	-	-	-
safeCleanseRate	double	Fixed	0.001	-	-	-	-
safeSuccessProb	double	Fixed	0.5	-	-	-	-

Range Study Variable Assignments for Study *st_strongatt_real* in Project *CAV_Exp_v3* :

Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-

LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.3	-	-	-	-
attackReset	double	Fixed	2.0	-	-	-	-
attackSuccess	double	Fixed	0.75	-	-	-	-
enableHazard	short	Manual	[0, 1]	-	-	-	-
enableOLH	short	Manual	[0, 1]	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
hazCleanseRate	double	Fixed	0.001	-	-	-	-
hazSuccessProb	double	Fixed	0.2209	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-
olhCleanseRate	double	Fixed	0.001	-	-	-	-
olhSuccessProb	double	Fixed	0.2209	-	-	-	-
safeCleanseRate	double	Fixed	0.001	-	-	-	-
safeSuccessProb	double	Fixed	0.2209	-	-	-	-

Range Study Variable Assignments for Study *st_weakatt* in Project *CAV_Exp_v3* :

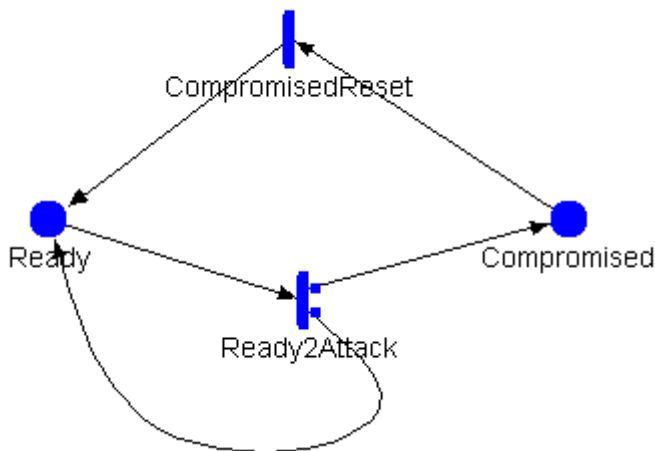
Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.1	-	-	-	-
attackReset	double	Fixed	2.0	-	-	-	-
attackSuccess	double	Fixed	0.25	-	-	-	-
enableHazard	short	Manual	[0, 1]	-	-	-	-
enableOLH	short	Manual	[0, 1]	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
hazCleanseRate	double	Fixed	0.001	-	-	-	-
hazSuccessProb	double	Fixed	0.3	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-
olhCleanseRate	double	Fixed	0.001	-	-	-	-

olhSuccessProb	double	Fixed	0.1	-	-	-	-
safeCleanseRate	double	Fixed	0.001	-	-	-	-
safeSuccessProb	double	Fixed	0.5	-	-	-	-
Range Study Variable Assignments for Study <i>st_weakatt_real</i> in Project <i>CAV_Exp_v3</i> :							
Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.1	-	-	-	-
attackReset	double	Fixed	2.0	-	-	-	-
attackSuccess	double	Fixed	0.25	-	-	-	-
enableHazard	short	Manual	[0, 1]	-	-	-	-
enableOLH	short	Manual	[0, 1]	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
hazCleanseRate	double	Fixed	0.001	-	-	-	-
hazSuccessProb	double	Fixed	0.2209	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-
olhCleanseRate	double	Fixed	0.001	-	-	-	-
olhSuccessProb	double	Fixed	0.2209	-	-	-	-
safeCleanseRate	double	Fixed	0.001	-	-	-	-
safeSuccessProb	double	Fixed	0.2209	-	-	-	-

HTML generated by Möbius using the “Document” feature.

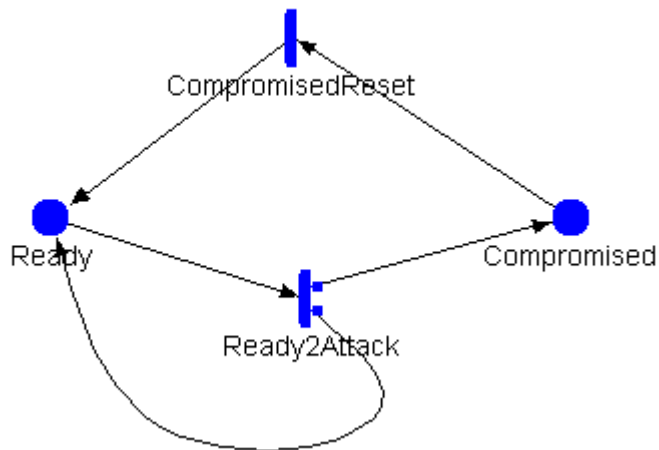
13.5 Uncertainty Model (Lomax and Exponential Distribution)

Model: AttackExponential (Atomic Model)



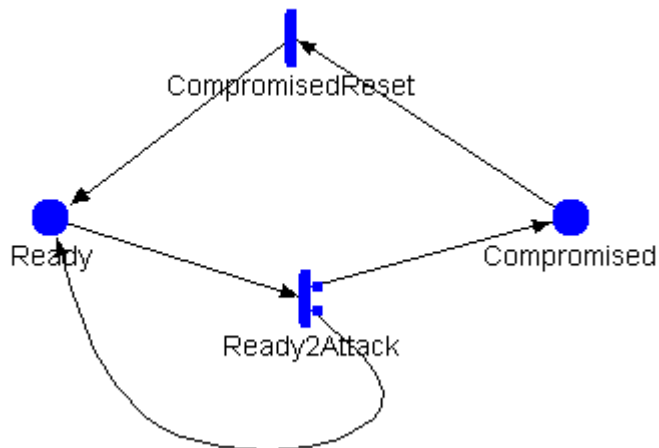
Place Attributes:	
Place Names	Initial Markings
Compromised	0
Ready	1
Timed Activity:	CompromisedReset
Distribution Parameters	Rate return(attackReset);
Activation Predicate	(none)
Reactivation Predicate	(none)
Timed Activity:	Ready2Attack
Distribution Parameters	Rate return(attackRate);
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 return(attackSuccess); case 2 return(1.0 - attackSuccess);

Model: AttackExponential2 (Atomic Model)



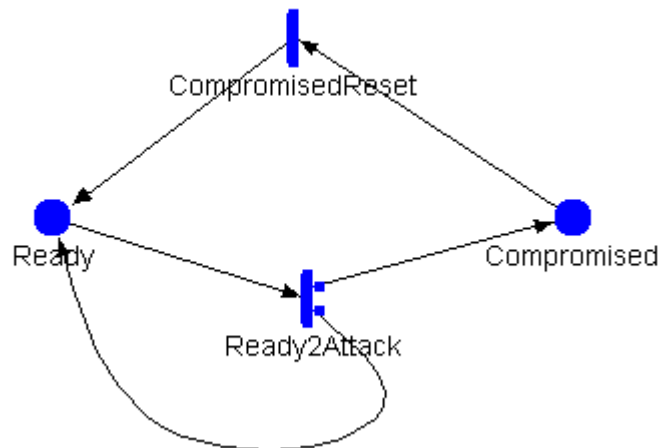
Place Attributes:	
Place Names	Initial Markings
Compromised	0
Ready	1
Timed Activity:	CompromisedReset
Distribution Parameters	Rate return(attackReset);
Activation Predicate	(none)
Reactivation Predicate	(none)
Timed Activity:	Ready2Attack
Distribution Parameters	Value static UserDistributions dist(0,31415); double lambda = attackRate; double delay; delay = dist.Exponential(lambda); return(delay);
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 return(attackSuccess); case 2 return(1.0 - attackSuccess);

Model: AttackLomax (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
Compromised	0
Ready	1
Timed Activity:	CompromisedReset
Distribution Parameters	Rate return(attackReset);
Activation Predicate	(none)
Reactivation Predicate	(none)
Timed Activity:	Ready2Attack
Distribution Parameters	Value static UserDistributions dist(0,31415); double lambda = attackRate; double delay; double xm; xm = (alpha - 1) / lambda; delay = dist.Pareto(xm,alpha) - xm; return(delay);
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 return(attackSuccess); case 2 return(1.0 - attackSuccess);

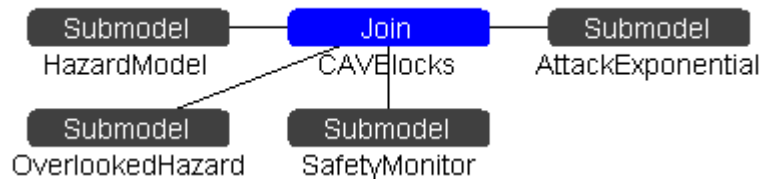
Model: AttackModel (Atomic Model)



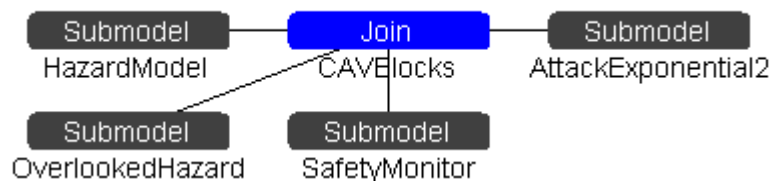
Place Attributes:	
Place Names	Initial Markings
Compromised	0
Ready	1
Timed Activity:	CompromisedReset
Distribution Parameters	Value static UserDistributions dist(0,31415); double lambda = attackReset; double delay; double xm; if (flag_CompromisedReset > 0) { xm = (alpha_CompromisedReset - 1) / lambda; delay = dist.Pareto(xm,alpha_CompromisedReset) - xm; } else { delay = dist.Exponential(lambda); } return(delay);
Activation Predicate	(none)
Reactivation Predicate	(none)
Timed Activity:	Ready2Attack
Distribution Parameters	Value static UserDistributions dist(0,31415); double lambda = attackRate; double delay; double xm; if (flag_Ready2Attack > 0) { xm = (alpha_Ready2Attack - 1) / lambda; delay = dist.Pareto(xm,alpha_Ready2Attack) - xm; } else { delay = dist.Exponential(lambda); } return(delay);
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1

	<pre> return(attackSuccess); case 2 return(1.0 - attackSuccess); </pre>
--	--

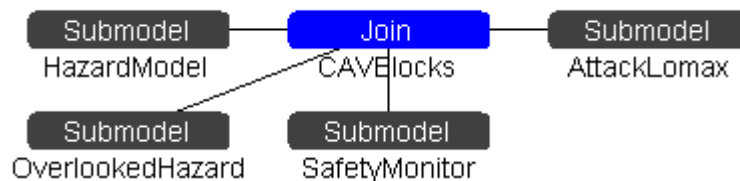
Model: CAV_ExponentialAttack (Composed Model)



Model: CAV_ExponentialAttack2 (Composed Model)



Model: CAV_LomaxAttack (Composed Model)



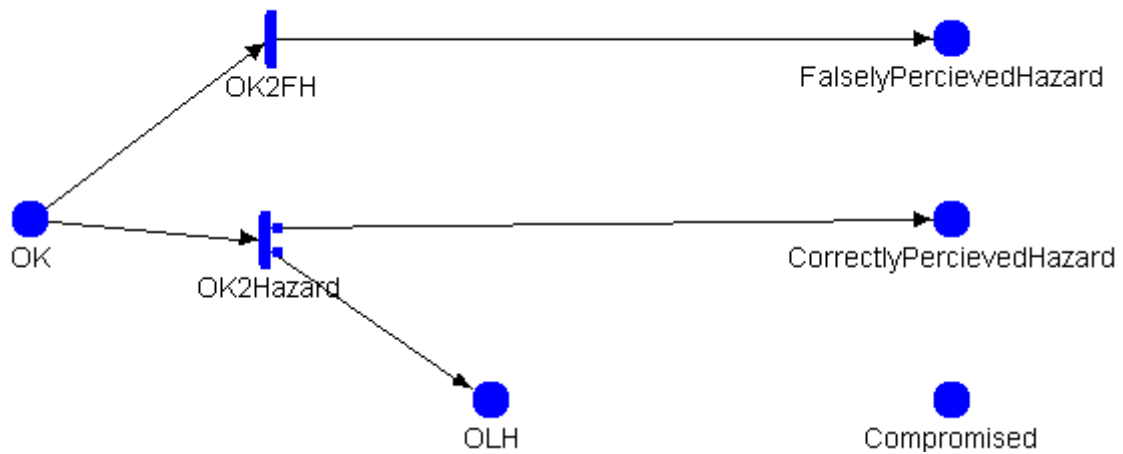
Performance Variable Model: ExponentialAccident		
Top Level Model Information	Child Model Name	CAV_ExponentialAttack
	Model Type	Rep/Join

Performance Variable : E_TimeToAccident		
Affecting Models	SafetyMonitor	
Impulse Functions		
Reward Function	<i>(Reward is over all Available Models)</i> <pre> if(SafetyMonitor->Accident->Mark() > 0) return(1); else return(0); </pre>	
Simulator Statistics	Type	Instant of Time
	Options	Estimate Mean
		Estimate Variance
		Include Lower Bound on Interval Estimate
		Include Upper Bound on Interval Estimate
		Estimate Distribution
		Estimate out of Range Probabilities
		Confidence Level is Relative

	Parameters	Start Time	100.0,1100.0,2100.0,3100.0,4100.0,5100.0,6100.0,7100.0,8100.0,9100.0,									
	Distributions	Lower Bound on Distribution Estimate	0.0									
		Upper Bound on Distribution Estimate	1.0									
		Step Size on Distribution Estimate	0.1									
	Confidence	Confidence Level	0.95									
		Confidence Interval	0.05									
Performance Variable Model: ExponentialAccident2												
Top Level Model Information		Child Model Name	CAV_ExponentialAttack2									
		Model Type	Rep/Join									
Performance Variable : E_TimeToAccident												
Affecting Models	SafetyMonitor											
Impulse Functions												
Reward Function	(Reward is over all Available Models) if(SafetyMonitor->Accident->Mark() > 0) return(1); else return(0);											
Simulator Statistics	Type	Instant of Time										
	Options	Estimate Mean										
		Estimate Variance										
		Include Lower Bound on Interval Estimate										
		Include Upper Bound on Interval Estimate										
		Estimate Distribution										
		Estimate out of Range Probabilities										
		Confidence Level is Relative										
	Parameters	Start Time	100.0,1100.0,2100.0,3100.0,4100.0,5100.0,6100.0,7100.0,8100.0,9100.0,									
	Distributions	Lower Bound on Distribution Estimate	0.0									
		Upper Bound on Distribution Estimate	1.0									
		Step Size on Distribution Estimate	0.1									
	Confidence	Confidence Level	0.95									

		Confidence Interval	0.05
--	--	---------------------	------

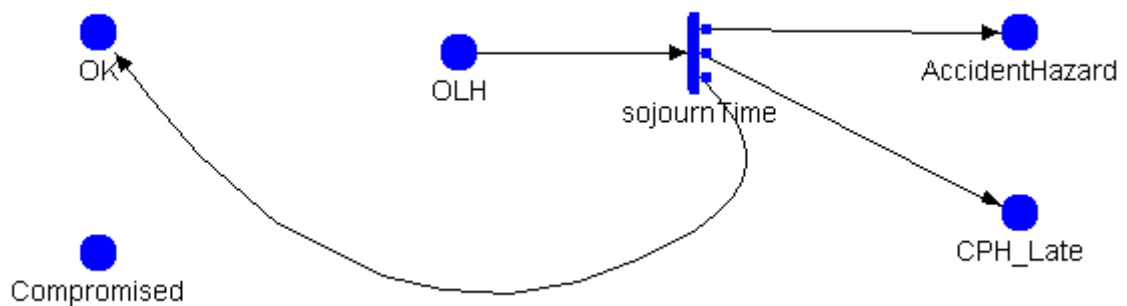
Model: HazardModel (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
Compromised	0
CorrectlyPercievedHazard	0
FalselyPercievedHazard	0
OK	1
OLH	0
Timed Activity:	OK2FH
Distribution Parameters	Rate return(HazardRate * falseHazardProb);
Activation Predicate	(none)
Reactivation Predicate	(none)
Timed Activity:	OK2Hazard
Distribution Parameters	Rate return(HazardRate * (1.0 - falseHazardProb));
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 return(1.0 - missHazardProb * pow(HazardScale,Compromised->Mark()) && enableHazard)); case 2 return(missHazardProb * pow(HazardScale,Compromised->Mark()) && enableHazard));
Performance Variable Model: LomaxAccident	
Top Level Model Information	Child Model Name CAV_LomaxAttack
	Model Type Rep/Join
Performance Variable : L_TimeToAccident	

Affecting Models	SafetyMonitor		
Impulse Functions			
Reward Function	<i>(Reward is over all Available Models)</i> if(SafetyMonitor->Accident->Mark() > 0) return(1); else return(0);		
Simulator Statistics	Type	Instant of Time	
	Options	Estimate Mean	
		Estimate Variance	
		Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate Distribution	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	100.0,1100.0,2100.0,3100.0,4100.0,5100.0,6100.0,7100.0,8100.0,9100.0,
	Distributions	Lower Bound on Distribution Estimate	0.0
		Upper Bound on Distribution Estimate	1.0
		Step Size on Distribution Estimate	0.1
	Confidence	Confidence Level	0.95
		Confidence Interval	0.05

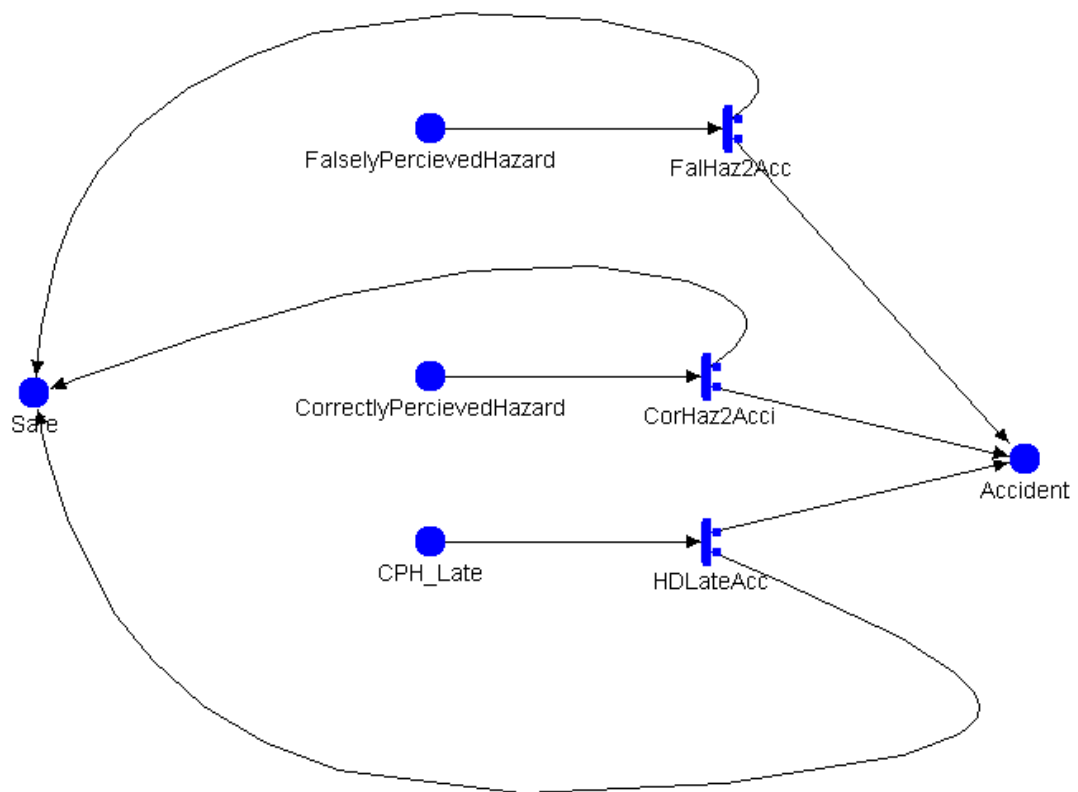
Model: OverlookedHazard (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
AccidentHazard	0
CPH_Late	0
Compromised	0

OK	1
OLH	0
Timed Activity:	sojournTime
Distribution Parameters	Rate return(OH2OKRate + MH2CHRate * pow(OH2CH_scale,Compromised->Mark() && enableOLH) + MH2AccRate * pow(DelayScale,Compromised->Mark() && enableOLH));
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 return((MH2AccRate * pow(DelayScale,Compromised->Mark() && enableOLH))/(OH2OKRate + MH2CHRate * pow(OH2CH_scale,Compromised->Mark() && enableOLH) + MH2AccRate * pow(DelayScale,Compromised->Mark() && enableOLH))); case 2 return((MH2CHRate * pow(OH2CH_scale,Compromised->Mark() && enableOLH))/(OH2OKRate + MH2CHRate * pow(OH2CH_scale,Compromised->Mark() && enableOLH) + MH2AccRate * pow(DelayScale,Compromised->Mark() && enableOLH))); case 3 return((OH2OKRate)/(OH2OKRate + MH2CHRate * pow(OH2CH_scale,Compromised->Mark() && enableOLH) + MH2AccRate * pow(DelayScale,Compromised->Mark() && enableOLH)));

Model: SafetyMonitor (Atomic Model)



isRSSperfect - a boolean variable:

1: if RSS is perfect (detected hazard cannot lead to accident)

0: if RSS is not perfect, then the hazard can lead to a an accident.

 Compromised

Place Attributes:	
Place Names	Initial Markings
Accident	0
CPH_Late	0
Compromised	0
CorrectlyPercievedHazard	0
FalselyPercievedHazard	0
Safe	1
Timed Activity:	CorHaz2Acci
Distribution Parameters	Rate return((CH2AccRate * pow(AccidentScale,Compromised->Mark()) && enableSM))*(1 - isRSSperfect) + CH2OKRate);
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1

	return(CH2OKRate/(((CH2AccRate * pow(AccidentScale,Compromised->Mark() && enableSM))*(1 - isRSSperfect) + CH2OKRate)); case 2 return(((CH2AccRate * pow(AccidentScale,Compromised->Mark() && enableSM))*(1 - isRSSperfect)/(((CH2AccRate * pow(AccidentScale,Compromised->Mark() && enableSM))*(1 - isRSSperfect) + CH2OKRate)));						
Timed Activity:	FalHaz2Acc						
Distribution Parameters	Rate return(FH2AccRate + FH2OKRate);						
Activation Predicate	(none)						
Reactivation Predicate	(none)						
Case Distributions	case 1 return(FH2OKRate / (FH2AccRate * pow(AccidentScale,Compromised->Mark() && enableSM) + FH2OKRate)); case 2 return(FH2AccRate * pow(AccidentScale,Compromised->Mark() && enableSM) / (FH2AccRate * pow(AccidentScale,Compromised->Mark() && enableSM) + FH2OKRate));						
Timed Activity:	HDLateAcc						
Distribution Parameters	Rate return(((CH2AccRate * pow(LateCH2AccScale,Compromised->Mark() && enableSM))*(1 - isRSSperfect) + CH2OKRate);						
Activation Predicate	(none)						
Reactivation Predicate	(none)						
Case Distributions	case 1 return((((CH2AccRate * pow(LateCH2AccScale,Compromised->Mark() && enableSM))*(1 - isRSSperfect))/((CH2AccRate * pow(LateCH2AccScale,Compromised->Mark() && enableSM))*(1 - isRSSperfect) + CH2OKRate)); case 2 return(CH2OKRate/(((CH2AccRate * pow(LateCH2AccScale,Compromised->Mark() && enableSM))*(1 - isRSSperfect) + CH2OKRate));						
Range Study Variable Assignments for Study <i>st_exponential</i> in Project <i>CAV_LomExpTest</i> :							
Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-

MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.1	-	-	-	-
attackReset	double	Fixed	2.0	-	-	-	-
attackSuccess	double	Fixed	0.2	-	-	-	-
enableHazard	short	Manual	[0, 1]	-	-	-	-
enableOLH	short	Manual	[0, 1]	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-

Range Study Variable Assignments for Study *st_exponential2* in Project *CAV_LomExpTest* :

Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
attackRate	double	Fixed	0.1	-	-	-	-
attackReset	double	Fixed	2.0	-	-	-	-
attackSuccess	double	Fixed	0.2	-	-	-	-
enableHazard	short	Manual	[0, 1]	-	-	-	-
enableOLH	short	Manual	[0, 1]	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-

Range Study Variable Assignments for Study *st_lomax_a10* in Project *CAV_LomExpTest* :

Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-

HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
alpha	double	Fixed	10.0	-	-	-	-
attackRate	double	Fixed	0.1	-	-	-	-
attackReset	double	Fixed	2.0	-	-	-	-
attackSuccess	double	Fixed	0.2	-	-	-	-
enableHazard	short	Manual	[0, 1]	-	-	-	-
enableOLH	short	Manual	[0, 1]	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-

Range Study Variable Assignments for Study *st_lomax_a2* in Project *CAV_LomExpTest* :

Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
alpha	double	Fixed	2.0	-	-	-	-
attackRate	double	Fixed	0.1	-	-	-	-
attackReset	double	Fixed	2.0	-	-	-	-
attackSuccess	double	Fixed	0.2	-	-	-	-
enableHazard	short	Manual	[0, 1]	-	-	-	-
enableOLH	short	Manual	[0, 1]	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-

Range Study Variable Assignments for Study *st_lomax_a5* in Project *CAV_LomExpTest* :

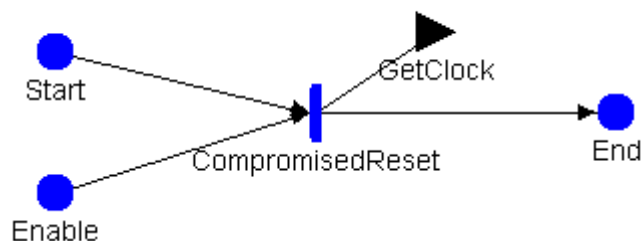
Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-

DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
alpha	double	Fixed	5.0	-	-	-	-
attackRate	double	Fixed	0.1	-	-	-	-
attackReset	double	Fixed	2.0	-	-	-	-
attackSuccess	double	Fixed	0.2	-	-	-	-
enableHazard	short	Manual	[0, 1]	-	-	-	-
enableOLH	short	Manual	[0, 1]	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-

HTML generated by Möbius using the “Document” feature.

13.6 Exponential Time Intervals Model

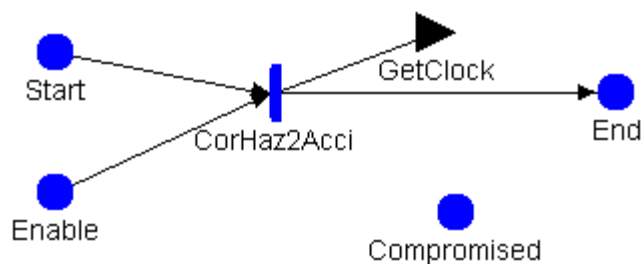
Model: CompromisedReset (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
Enable	1
End	0
Start	10000
Timed Activity:	CompromisedReset
Distribution Parameters	Rate return(attackReset);
Activation Predicate	(none)
Reactivation Predicate	(none)
Output Gate:	GetClock
Function	#include <iostream>

	using namespace std; float t = BaseModelClass::LastActionTime; cout << t << endl; Enable->Mark()=1;		
Performance Variable Model: CompromisedReset			
Top Level Model Information	Child Model Name	CompromisedReset	
	Model Type	SAN Model	
Performance Variable : Time			
Affecting Models	CompromisedReset		
Impulse Functions			
Reward Function	(Reward is over all Available Models) if (CompromisedReset->Start->Mark() > 0) return(1); else return(0);		
Simulator Statistics	Type	Instant of Time	
	Options	Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	1000.0,2000.0,3000.0,4000.0,5000.0,6000.0,7000.0,8000.0,9000.0,10000.0,
Confidence	Confidence Level	0.95	
	Confidence Interval	0.1	
Set Study: CompromisedReset:			
Experiment	Variable	Type	Value
Experiment 1	attackReset	double	2.0

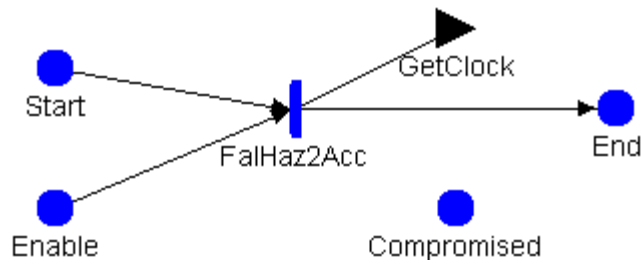
Model: CorHaz2Acci (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
Compromised	0
Enable	1
End	0
Start	10000
Timed Activity:	CorHaz2Acci

Distribution Parameters	Rate return((CH2AccRate * pow(AccidentScale,Compromised->Mark() && enableSM))*(1 - isRSSperfect) + CH2OKRate);						
Activation Predicate	(none)						
Reactivation Predicate	(none)						
Output Gate:	GetClock						
Function	#include <iostream> using namespace std; float t = BaseModelClass::LastActionTime; cout << t << endl; Enable->Mark()=1;						
Performance Variable Model: CorHaz2Acci							
Top Level Model Information	Child Model Name		CorHaz2Acci				
	Model Type		SAN Model				
Performance Variable : Time							
Affecting Models	CorHaz2Acci						
Impulse Functions							
Reward Function	<i>(Reward is over all Available Models)</i> if (CorHaz2Acci->Start->Mark() > 0) return(1); else return(0);						
Simulator Statistics	Type	Instant of Time					
	Options	Include Lower Bound on Interval Estimate					
		Include Upper Bound on Interval Estimate					
		Estimate out of Range Probabilities					
		Confidence Level is Relative					
	Parameters	Start Time	1000.0,				
	Confidence	Confidence Level	0.95				
Confidence Interval		0.1					
Range Study Variable Assignments for Study <i>CorHaz2Acci</i> in Project <i>T_Exponential</i> :							
Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-

Model: FalHaz2Acc (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
Compromised	0
Enable	1
End	0
Start	10000
Timed Activity:	FalHaz2Acc
Distribution Parameters	Rate return(FH2AccRate + FH2OKRate);
Activation Predicate	(none)
Reactivation Predicate	(none)
Output Gate:	GetClock
Function	<pre> #include <iostream> using namespace std; float t = BaseModelClass::LastActionTime; cout << t << endl; Enable->Mark()=1; </pre>

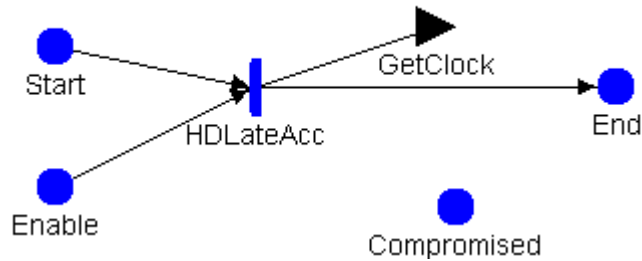
Performance Variable Model: FalHaz2Acc		
Top Level Model Information	Child Model Name	FalHaz2Acc
	Model Type	SAN Model

Performance Variable : Time			
Affecting Models	FalHaz2Acc		
Impulse Functions			
Reward Function	<i>(Reward is over all Available Models)</i> if (FalHaz2Acc->Start->Mark() > 0) return(1); else return(0);		
Simulator Statistics	Type	Instant of Time	
	Options	Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	1000.0,
	Confidence	Confidence Level	0.95
		Confidence Interval	0.1

Set Study: FalHaz2Acc:

Experiment	Variable	Type	Value
Experiment 1	FH2AccRate	double	1.0E-5
	FH2OKRate	double	856.3

Model: HDLateAcc (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
Compromised	0
Enable	1
End	0
Start	10000
Timed Activity:	HDLateAcc
Distribution Parameters	Rate return((CH2AccRate * pow(LateCH2AccScale,Compromised->Mark() && enableSM))*(1 - isRSSperfect) + CH2OKRate);
Activation Predicate	(none)
Reactivation Predicate	(none)
Output Gate:	GetClock
Function	<pre>#include <iostream> using namespace std; float t = BaseModelClass::LastActionTime; cout << t << endl; Enable->Mark()=1;</pre>

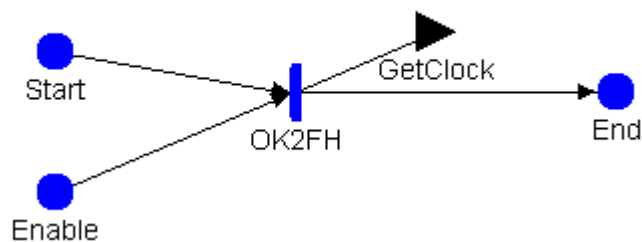
Performance Variable Model: HDLateAcc		
Top Level Model Information	Child Model Name	HDLateAcc
	Model Type	SAN Model

Performance Variable : Time		
Affecting Models	HDLateAcc	
Impulse Functions		
Reward Function	<i>(Reward is over all Available Models)</i> if (HDLateAcc->Start->Mark() > 0) return(1); else return(0);	
Simulator Statistics	Type	Instant of Time
	Options	Include Lower Bound on Interval Estimate

		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	1000.0,
	Confidence	Confidence Level	0.95
		Confidence Interval	0.1

Set Study: HDLateAcc:			
Experiment	Variable	Type	Value
Experiment 1	CH2AccRate	double	1.0E-4
	CH2OKRate	double	856.3
	LateCH2AccScale	double	10.0
	enableSM	short	0
	isRSSperfect	int	0
Experiment 2	CH2AccRate	double	1.0E-4
	CH2OKRate	double	856.3
	LateCH2AccScale	double	10.0
	enableSM	short	1
	isRSSperfect	int	0

Model: OK2FH (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
Enable	1
End	0
Start	10000
Timed Activity:	OK2FH
Distribution Parameters	Rate return(HazardRate * falseHazardProb);
Activation Predicate	(none)
Reactivation Predicate	(none)
Output Gate:	GetClock
Function	<pre>#include <iostream> using namespace std; float t = BaseModelClass::LastActionTime; cout << t << endl; Enable->Mark()=1;</pre>

Performance Variable Model: OK2FH
--

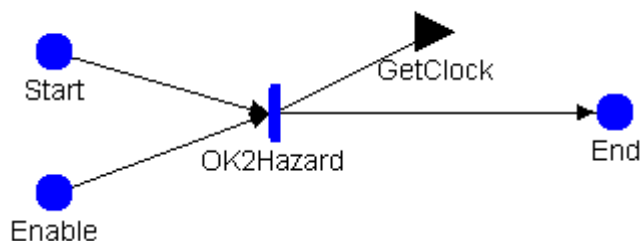
Top Level Model Information	Child Model Name	OK2FH
	Model Type	SAN Model

Performance Variable : Time			
A ff ec ti ng M od el s	OK2FH		
I m pu ls e F un cti on s			
R e w ar d F un cti on	<i>(Reward is over all Available Models)</i> if (OK2FH->Start->Mark() > 0) return(1); else return(0);		
Si m ul at or St ati sti cs	Typ	Instant of Time	
	Opti on s	O	Include Lower Bound on Interval Estimate
		pti	Include Upper Bound on Interval Estimate
		on	Estimate out of Range Probabilities
		s	Confidence Level is Relative
	Pa ra m ete rs	St art Ti me	1000.0,2000.0,3000.0,4000.0,5000.0,6000.0,7000.0,8000.0,9000.0,10000.0,11000.0,12000.0,13000.0,14000.0,15000.0,16000.0,17000.0,18000.0,19000.0,20000.0,21000.0,22000.0,23000.0,24000.0,25000.0,26000.0,27000.0,28000.0,29000.0,30000.0,31000.0,32000.0,33000.0,34000.0,35000.0,36000.0,37000.0,38000.0,39000.0,40000.0,41000.0,42000.0,43000.0,44000.0,45000.0,46000.0,47000.0,48000.0,49000.0,50000.0,51000.0,52000.0,53000.0,54000.0,55000.0,56000.0,57000.0,58000.0,59000.0,60000.0,61000.0,62000.0,63000.0,64000.0,65000.0,66000.0,67000.0,68000.0,69000.0,70000.0,71000.0,72000.0,73000.0,74000.0,75000.0,76000.0,77000.0,78000.0,79000.0,80000.0,81000.0,82000.0,83000.0,84000.0,85000.0,86000.0,87000.0,88000.0,89000.0,90000.0,91000.0,92000.0,93000.0,94000.0,95000.0,96000.0,97000.0,98000.0,99000.0,100000.0,101000.0,102000.0,103000.0,104000.0,105000.0,106000.0,107000.0,108000.0,109000.0,110000.0,111000.0,112000.0,113000.0,114000.0,115000.0,116000.0,117000.0,118000.0,119000.0,120000.0,121000.0,122000.0,123000.0,124000.0,125000.0,126000.0,127000.0,128000.0,129000.0,130000.0,131000.0,132000.0,133000.0,134000.0,135000.0,136000.0,137000.0,138000.0,139000.0,140000.0,141000.0,142000.0,143000.0,144000.0,145000.0,146000.0,147000.0,148000.0,149000.0,150000.0,151000.0,152000.0,153000.0,154000.0,155000.0,156000.0,157000.0,158000.0,159000.0,160000.0,161000.0,162000.0,163000.0,164000.0,165000.0,166000.0,167000.0,168000.0,169000.0,170000.0,171000.0,172000.0,173000.0,174000.0,175000.0,176000.0,177000.0,178000.0,179000.0,180000.0,181000.0,182000.0,183000.0,184000.0,185000.0,186000.0,187000.0,188000.0,189000.0,190000.0,191000.0,192000.0,193000.0,194000.0,195000.0,196000.0,197000.0,198000.0,199000.0,200000.0,201000.0,202000.0,203000.0,204000.0,205000.0,206000.0,207000.0,208000.0,209000.0,210000.0,211000.0,212000.0,213000.0,214000.0,

		815000.0,816000.0,817000.0,818000.0,819000.0,820000.0,821000.0,822000.0,823000.0,824000.0,825000.0,826000.0,827000.0,828000.0,829000.0,830000.0,831000.0,832000.0,833000.0,834000.0,835000.0,836000.0,837000.0,838000.0,839000.0,840000.0,841000.0,842000.0,843000.0,844000.0,845000.0,846000.0,847000.0,848000.0,849000.0,850000.0,851000.0,852000.0,853000.0,854000.0,855000.0,856000.0,857000.0,858000.0,859000.0,860000.0,861000.0,862000.0,863000.0,864000.0,865000.0,866000.0,867000.0,868000.0,869000.0,870000.0,871000.0,872000.0,873000.0,874000.0,875000.0,876000.0,877000.0,878000.0,879000.0,880000.0,881000.0,882000.0,883000.0,884000.0,885000.0,886000.0,887000.0,888000.0,889000.0,890000.0,891000.0,892000.0,893000.0,894000.0,895000.0,896000.0,897000.0,898000.0,899000.0,900000.0,901000.0,902000.0,903000.0,904000.0,905000.0,906000.0,907000.0,908000.0,909000.0,910000.0,911000.0,912000.0,913000.0,914000.0,915000.0,916000.0,917000.0,918000.0,919000.0,920000.0,921000.0,922000.0,923000.0,924000.0,925000.0,926000.0,927000.0,928000.0,929000.0,930000.0,931000.0,932000.0,933000.0,934000.0,935000.0,936000.0,937000.0,938000.0,939000.0,940000.0,941000.0,942000.0,943000.0,944000.0,945000.0,946000.0,947000.0,948000.0,949000.0,950000.0,951000.0,952000.0,953000.0,954000.0,955000.0,956000.0,957000.0,958000.0,959000.0,960000.0,961000.0,962000.0,963000.0,964000.0,965000.0,966000.0,967000.0,968000.0,969000.0,970000.0,971000.0,972000.0,973000.0,974000.0,975000.0,976000.0,977000.0,978000.0,979000.0,980000.0,981000.0,982000.0,983000.0,984000.0,985000.0,986000.0,987000.0,988000.0,989000.0,990000.0,991000.0,992000.0,993000.0,994000.0,995000.0,996000.0,997000.0,998000.0,999000.0,1000000.0,
Co nfi de nc e	Co nfi de nc e	0.95
Co nfi de nc e	Co nfi de nc e	0.1
	Le ve l	Int er va l

Set Study: OK2FH :			
Experiment	Variable	Type	Value
Experiment 1	HazardRate	Double	197.4
	falseHazardProb	Double	1.0E-4

Model: OK2Hazard (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
Enable	1
End	0

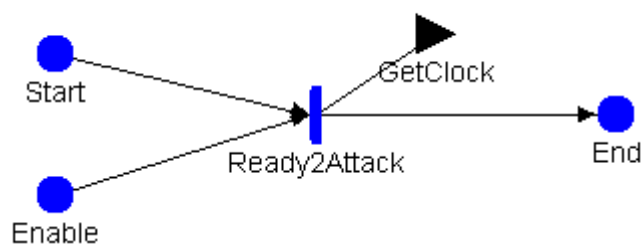
Start	10000
Timed Activity:	OK2Hazard
Distribution Parameters	Rate return(HazardRate * (1.0 - falseHazardProb));
Activation Predicate	(none)
Reactivation Predicate	(none)
Output Gate:	GetClock
Function	#include <iostream> using namespace std; float t = BaseModelClass::LastActionTime; cout << t << endl; Enable->Mark()=1;

Performance Variable Model: OK2Hazard		
Top Level Model Information	Child Model Name	OK2Hazard
	Model Type	SAN Model

Performance Variable : Time			
Affecting Models	OK2Hazard		
Impulse Functions			
Reward Function	(Reward is over all Available Models) if (OK2Hazard->Start->Mark() > 0) return(1); else return(0);		
Simulator Statistics	Type	Instant of Time	
	Options	Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	1000.0,
	Confidence	Confidence Level	0.95
		Confidence Interval	0.1

Set Study: OK2Hazard:			
Experiment	Variable	Type	Value
Experiment 1	HazardRate	double	197.4
	falseHazardProb	double	1.0E-4

Model: Ready2Attack (Atomic Model)



Place Attributes:

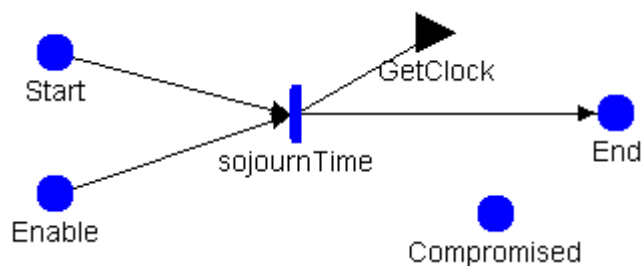
Place Names		Initial Markings	
Enable		1	
End		0	
Start		10000	
Timed Activity:		Ready2Attack	
Distribution Parameters		Rate return(attackRate);	
Activation Predicate		(none)	
Reactivation Predicate		(none)	
Output Gate:		GetClock	
Function		#include <iostream> using namespace std; float t = BaseModelClass::LastActionTime; cout << t << endl; Enable->Mark()=1;	
Performance Variable Model: Ready2Attack			
Top Level Model Information		Child Model Name	Ready2Attack
		Model Type	SAN Model
Performance Variable : Time			
A ff ec ti ng M od el s	Ready2Attack		
I m pu ls e F un cti on s			
R e w ar d F un cti on	(Reward is over all Available Models) if (Ready2Attack->Start->Mark() > 0) return(1); else return(0);		
Sim ul at or	Type	Instant of Time	
	Opti	Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	

Statistics	Confidence Level is Relative		
Parameters	Start Time	1000.0,2000.0,3000.0,4000.0,5000.0,6000.0,7000.0,8000.0,9000.0,10000.0,11000.0,12000.0,13000.0,14000.0,15000.0,16000.0,17000.0,18000.0,19000.0,20000.0,21000.0,22000.0,23000.0,24000.0,25000.0,26000.0,27000.0,28000.0,29000.0,30000.0,31000.0,32000.0,33000.0,34000.0,35000.0,36000.0,37000.0,38000.0,39000.0,40000.0,41000.0,42000.0,43000.0,44000.0,45000.0,46000.0,47000.0,48000.0,49000.0,50000.0,51000.0,52000.0,53000.0,54000.0,55000.0,56000.0,57000.0,58000.0,59000.0,60000.0,61000.0,62000.0,63000.0,64000.0,65000.0,66000.0,67000.0,68000.0,69000.0,70000.0,71000.0,72000.0,73000.0,74000.0,75000.0,76000.0,77000.0,78000.0,79000.0,80000.0,81000.0,82000.0,83000.0,84000.0,85000.0,86000.0,87000.0,88000.0,89000.0,90000.0,91000.0,92000.0,93000.0,94000.0,95000.0,96000.0,97000.0,98000.0,99000.0,100000.0,	
Confidence Level	Confidence	0.95	
Confidence	Confidence Interval	0.1	

Set Study: [Ready2Attack:](#)

Experiment	Variable	Type	Value
Experiment 1	attackRate	double	0.1

Model: sojournTime (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
Compromised	1
Enable	1
End	0
Start	10000
Timed Activity:	sojournTime
Distribution Parameters	Rate $\text{return}(\text{OH2OKRate} + \text{MH2CHRate} * \text{pow}(\text{OH2CH_scale}, \text{Compromised} \rightarrow \text{Mark}()) \&\& \text{enableOLH}) + \text{MH2AccRate} * \text{pow}(\text{DelayScale}, \text{Compromised} \rightarrow \text{Mark}()) \&\& \text{enableOLH});$

Activation Predicate	(none)
Reactivation Predicate	(none)
Output Gate:	GetClock
Function	<pre>#include <iostream> using namespace std; float t = BaseModelClass::LastActionTime; cout << t << endl; Enable->Mark()=1;</pre>

Performance Variable Model: sojournTime		
Top Level Model Information	Child Model Name	sojournTime
	Model Type	SAN Model

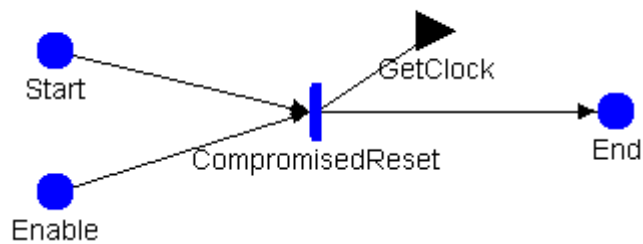
Performance Variable : Time			
Affecting Models	sojournTime		
Impulse Functions			
Reward Function	<i>(Reward is over all Available Models)</i> if (sojournTime->Start->Mark() > 0) return(1); else return(0);		
Simulator Statistics	Type	Instant of Time	
	Options	Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	1000.0,
	Confidence	Confidence Level	0.95
		Confidence Interval	0.1

Range Study Variable Assignments for Study <i>sojournTime</i> in Project <i>T_Exponential</i> :							
Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
DelayScale	double	Fixed	1000.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
enableOLH	short	Manual	[0, 1]	-	-	-	-

HTML generated by Möbius using the “Document” feature.

13.7 Exponential Time Intervals Model (Deterministic)

Model: CompromisedReset (Atomic Model)



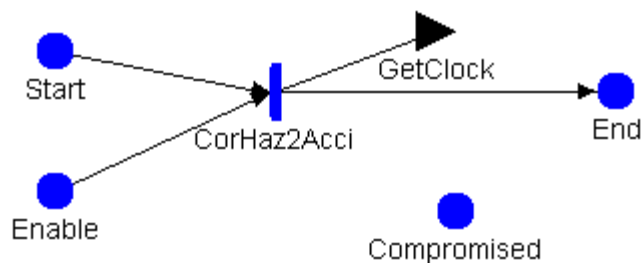
Place Attributes:		
Place Names		Initial Markings
Enable		1
End		0
Start		10000
Timed Activity:		CompromisedReset
Distribution Parameters	Value static UserDistributions dist(0,31415); double lambda = attackReset; double delay = dist.Exponential(lambda); return(delay);	
Activation Predicate		(none)
Reactivation Predicate		(none)
Output Gate:		GetClock
Function	#include <iostream> using namespace std; float t = BaseModelClass::LastActionTime; cout << t << endl; Enable->Mark()=1;	
Performance Variable Model: CompromisedReset		
Top Level Model Information	Child Model Name	CompromisedReset
	Model Type	SAN Model
Performance Variable : Time		
Affecting Models	CompromisedReset	
Impulse Functions		
Reward Function	(Reward is over all Available Models) if (CompromisedReset->Start->Mark() > 0) return(1); else return(0);	
Simulator Statistics	Type	Instant of Time
	Options	Include Lower Bound on Interval Estimate
		Include Upper Bound on Interval Estimate
		Estimate out of Range Probabilities
		Confidence Level is Relative

	Parameters	Start Time	1000.0,2000.0,3000.0,4000.0,5000.0,6000.0,7000.0,8000.0,9000.0,10000.0,
	Confidence	Confidence Level	0.95
		Confidence Interval	0.1

Set Study: **CompromisedReset:**

Experiment	Variable	Type	Value
Experiment 1	attackReset	double	2.0

Model: CorHaz2Acci (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
Compromised	0
Enable	1
End	0
Start	10000

Timed Activity:	CorHaz2Acci
Distribution Parameters	Value static UserDistributions dist(0,31415); double lambda = ((CH2AccRate * pow(AccidentScale,Compromised->Mark()) && enableSM))*(1 - isRSSperfect) + CH2OKRate); double delay = dist.Exponential(lambda); return(delay);
Activation Predicate	(none)
Reactivation Predicate	(none)

Output Gate:	GetClock
Function	<pre> #include <iostream> using namespace std; float t = BaseModelClass::LastActionTime; cout << t << endl; Enable->Mark()=1; </pre>

Performance Variable Model: CorHaz2Acci		
Top Level Model Information	Child Model Name	CorHaz2Acci
	Model Type	SAN Model

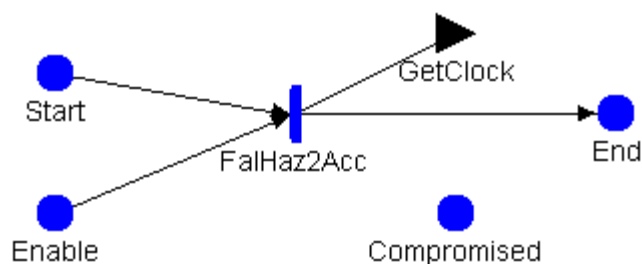
Performance Variable : Time	
Affecting Models	CorHaz2Acci
Impulse Functions	

Reward Function	(Reward is over all Available Models) if (CorHaz2Acci->Start->Mark() > 0) return(1); else return(0);	
Simulator Statistics	Type	Instant of Time
	Options	Include Lower Bound on Interval Estimate
		Include Upper Bound on Interval Estimate
		Estimate out of Range Probabilities
		Confidence Level is Relative
	Parameters	Start Time 1000.0,
	Confidence	Confidence Level 0.95
		Confidence Interval 0.1

Range Study Variable Assignments for Study *CorHaz2Acci* in Project *T_Exponential2* :

Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-

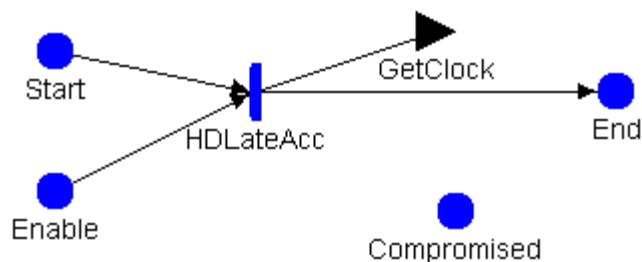
Model: FalHaz2Acc (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
Compromised	0
Enable	1
End	0
Start	10000
Timed Activity:	FalHaz2Acc
Distribution Parameters	Value static UserDistributions dist(0,31415); double lambda = (FH2AccRate + FH2OKRate); double delay = dist.Exponential(lambda); return(delay);
Activation Predicate	(none)
Reactivation Predicate	(none)
Output Gate:	GetClock
Function	#include <iostream>

	using namespace std; float t = BaseModelClass::LastActionTime; cout << t << endl; Enable->Mark()=1;		
Performance Variable Model: FalHaz2Acc			
Top Level Model Information	Child Model Name	FalHaz2Acc	
	Model Type	SAN Model	
Performance Variable : Time			
Affecting Models	FalHaz2Acc		
Impulse Functions			
Reward Function	(Reward is over all Available Models) if (FalHaz2Acc->Start->Mark() > 0) return(1); else return(0);		
Simulator Statistics	Type	Instant of Time	
	Options	Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	1000.0,
	Confidence	Confidence Level	0.95
Confidence Interval		0.1	
Set Study: FalHaz2Acc:			
Experiment	Variable	Type	Value
Experiment 1	FH2AccRate	double	1.0E-5
	FH2OKRate	double	856.3

Model: HDLateAcc (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
Compromised	0
Enable	1
End	0
Start	10000
Timed Activity:	HDLateAcc
Distribution Parameters	Value static UserDistributions dist(0,31415);

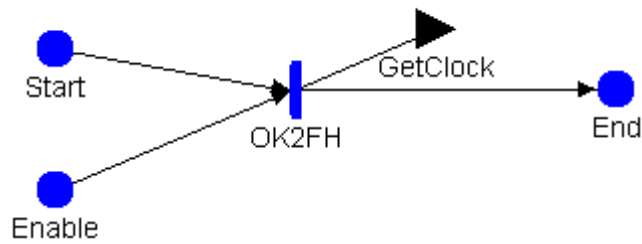
	double lambda = ((CH2AccRate * pow(LateCH2AccScale,Compromised->Mark() && enableSM))*(1 - isRSSperfect) + CH2OKRate); double delay = dist.Exponential(lambda); return(delay);
Activation Predicate	(none)
Reactivation Predicate	(none)
Output Gate:	GetClock
Function	<pre>#include <iostream> using namespace std; float t = BaseModelClass::LastActionTime; cout << t << endl; Enable->Mark()=1;</pre>

Performance Variable Model: HDLateAcc		
Top Level Model Information	Child Model Name	HDLateAcc
	Model Type	SAN Model

Performance Variable : Time			
Affecting Models	HDLateAcc		
Impulse Functions			
Reward Function	<i>(Reward is over all Available Models)</i> if (HDLateAcc->Start->Mark() > 0) return(1); else return(0);		
Simulator Statistics	Type	Instant of Time	
	Options	Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	1000.0,
	Confidence	Confidence Level	0.95
		Confidence Interval	0.1

Set Study: HDLateAcc:			
Experiment	Variable	Type	Value
Experiment 1	CH2AccRate	double	1.0E-4
	CH2OKRate	double	856.3
	LateCH2AccScale	double	10.0
	enableSM	short	0
	isRSSperfect	int	0
Experiment 2	CH2AccRate	double	1.0E-4
	CH2OKRate	double	856.3
	LateCH2AccScale	double	10.0
	enableSM	short	1
	isRSSperfect	int	0

Model: OK2FH (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
Enable	1
End	0
Start	10000
Timed Activity:	OK2FH
Distribution Parameters	Value static UserDistributions dist(0,31415); double lambda = (HazardRate * falseHazardProb); double delay = dist.Exponential(lambda); return(delay);
Activation Predicate	(none)
Reactivation Predicate	(none)
Output Gate:	GetClock
Function	#include <iostream> using namespace std; float t = BaseModelClass::LastActionTime; cout << t << endl; Enable->Mark()=1;

Performance Variable Model: OK2FH		
Top Level Model Information	Child Model Name	OK2FH
	Model Type	SAN Model

Performance Variable : Time	
Affecting Models	OK2FH
Impulse Function	

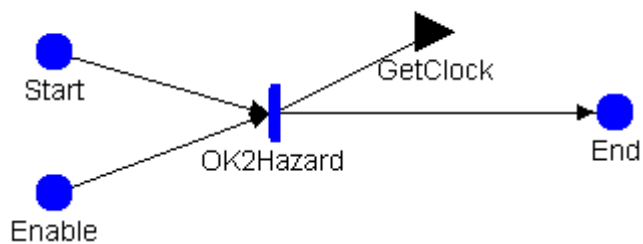
ons			
Reward Function	(Reward is over all Available Models) if (OK2FH->Start->Mark() > 0) return(1); else return(0);		
	Type	Instant of Time	
	Options	Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
Simulator Statistics	Parameters	Start Time	1000.0,2000.0,3000.0,4000.0,5000.0,6000.0,7000.0,8000.0,9000.0,10000.0,11000.0,12000.0,13000.0,14000.0,15000.0,16000.0,17000.0,18000.0,19000.0,20000.0,21000.0,22000.0,23000.0,24000.0,25000.0,26000.0,27000.0,28000.0,29000.0,30000.0,31000.0,32000.0,33000.0,34000.0,35000.0,36000.0,37000.0,38000.0,39000.0,40000.0,41000.0,42000.0,43000.0,44000.0,45000.0,46000.0,47000.0,48000.0,49000.0,50000.0,51000.0,52000.0,53000.0,54000.0,55000.0,56000.0,57000.0,58000.0,59000.0,60000.0,61000.0,62000.0,63000.0,64000.0,65000.0,66000.0,67000.0,68000.0,69000.0,70000.0,71000.0,72000.0,73000.0,74000.0,75000.0,76000.0,77000.0,78000.0,79000.0,80000.0,81000.0,82000.0,83000.0,84000.0,85000.0,86000.0,87000.0,88000.0,89000.0,90000.0,91000.0,92000.0,93000.0,94000.0,95000.0,96000.0,97000.0,98000.0,99000.0,100000.0,101000.0,102000.0,103000.0,104000.0,105000.0,106000.0,107000.0,108000.0,109000.0,110000.0,111000.0,112000.0,113000.0,114000.0,115000.0,116000.0,117000.0,118000.0,119000.0,120000.0,121000.0,122000.0,123000.0,124000.0,125000.0,126000.0,127000.0,128000.0,129000.0,130000.0,131000.0,132000.0,133000.0,134000.0,135000.0,136000.0,137000.0,138000.0,139000.0,140000.0,141000.0,142000.0,143000.0,144000.0,145000.0,146000.0,147000.0,148000.0,149000.0,150000.0,151000.0,152000.0,153000.0,154000.0,155000.0,156000.0,157000.0,158000.0,159000.0,160000.0,161000.0,162000.0,163000.0,164000.0,165000.0,166000.0,167000.0,168000.0,169000.0,170000.0,171000.0,172000.0,173000.0,174000.0,175000.0,176000.0,177000.0,178000.0,179000.0,180000.0,181000.0,182000.0,183000.0,184000.0,185000.0,186000.0,187000.0,188000.0,189000.0,190000.0,191000.0,192000.0,193000.0,194000.0,195000.0,196000.0,197000.0,198000.0,199000.0,200000.0,201000.0,202000.0,203000.0,204000.0,205000.0,206000.0,207000.0,208000.0,209000.0,210000.0,211000.0,212000.0,213000.0,214000.0,215000.0,216000.0,217000.0,218000.0,219000.0,220000.0,221000.0,222000.0,223000.0,224000.0,225000.0,226000.0,227000.0,228000.0,229000.0,230000.0,231000.0,232000.0,233000.0,234000.0,235000.0,236000.0,237000.0,238000.0,239000.0,240000.0,241000.0,242000.0,243000.0,244000.0,245000.0,246000.0,247000.0,248000.0,249000.0,250000.0,251000.0,252000.0,253000.0,254000.0,255000.0,256000.0,257000.0,258000.0,259000.0,260000.0,261000.0,262000.0,263000.0,264000.0,265000.0,266000.0,267000.0,268000.0,269000.0,270000.0,271000.0,272000.0,273000.0,274000.0,275000.0,276000.0,277000.0,278000.0,279000.0,280000.0,281000.0,282000.0,283000.0,284000.0,285000.0,286000.0,287000.0,288000.0,289000.0,290000.0,291000.0,292000.0,293000.0,294000.0,295000.0,296000.0,297000.0,298000.0,299000.0,300000.0,301000.0,302000.0,303000.0,304000.0,305000.0,306000.0,307000.0,308000.0,309000.0,310000.0,311000.0,312000.0,313000.0,314000.0,315000.0,316000.0,317000.0,318000.0,319000.0,320000.0,321000.0,322000.0,323000.0,324000.0,325000.0,326000.0,327000.0,328000.0,329000.0,330000.0,331000.0,332000.0,333000.0,334000.0,335000.0,336000.0,337000.0,338000.0,339000.0,340000.0,341000.0,342000.0,343000.0,344000.0,345000.0,346000.0,347000.0,348000.0,349000.0,350000.0,351000.0,352000.0,353000.0,354000.0,355000.0,356000.0,357000.0,358000.0,359000.0,360000.0,361000.0,362000.0,363000.0,364000.0,365000.0,366000.0,367000.0,368000.0,369000.0,370000.0,371000.0,372000.0,373000.0,374000.0,375000.0,376000.0,377000.0,378000.0,379000.0,380000.0,381000.0,382000.0,383000.0,384000.0,385000.0,386000.0,387000.0,388000.0,389000.0,390000.0,391000.0,392000.0,393000.0,394000.0,395000.0,396000.0,397000.0,398000.0,399000.0,400000.0,401000.0,402000.0,403000.0,404000.0,405000.0,406000.0,407000.0,408000.0,409000.0,410000.0,411000.0,412000.0,413000.0,414000.0,415000.0,416000.0,417000.0,418000.0,419000.0,420000.0,421000.0,422000.0,423000.0,424000.0,

[illegible]

de nc e	de nc e Le ve l	
	Co nfi de nc e Int er va l	0.1

Set Study: OK2FH :			
Experiment	Variable	Type	Value
Experiment 1	HazardRate	Double	197.4
	falseHazardProb	Double	1.0E-4

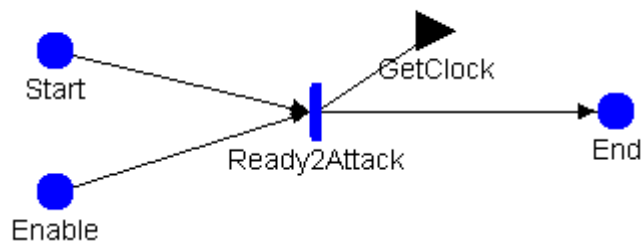
Model: OK2Hazard (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
Enable	1
End	0
Start	10000
Timed Activity:	OK2Hazard
Distribution Parameters	Value <pre> static UserDistributions dist(0,31415); double lambda = (HazardRate * (1.0 - falseHazardProb)); double delay = dist.Exponential(lambda); return(delay); </pre>
Activation Predicate	(none)
Reactivation Predicate	(none)
Output Gate:	GetClock
Function	<pre> #include <iostream> using namespace std; float t = BaseModelClass::LastActionTime; cout << t << endl; Enable->Mark()=1; </pre>

Performance Variable Model: OK2Hazard			
Top Level Model Information	Child Model Name	OK2Hazard	
	Model Type	SAN Model	
Performance Variable : Time			
Affecting Models	OK2Hazard		
Impulse Functions			
Reward Function	<i>(Reward is over all Available Models)</i> if (OK2Hazard->Start->Mark() > 0) return(1); else return(0);		
Simulator Statistics	Type	Instant of Time	
	Options	Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	1000.0,
	Confidence	Confidence Level	0.95
		Confidence Interval	0.1
Set Study: OK2Hazard:			
Experiment	Variable	Type	Value
Experiment 1	HazardRate	double	197.4
	falseHazardProb	double	1.0E-4

Model: Ready2Attack (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
Enable	1
End	0
Start	10000
Timed Activity:	Ready2Attack
Distribution Parameters	Value static UserDistributions dist(0,31415); double lambda = attackRate; double delay = dist.Exponential(lambda); return(delay);
Activation Predicate	(none)
Reactivation Predicate	(none)
Output Gate:	GetClock

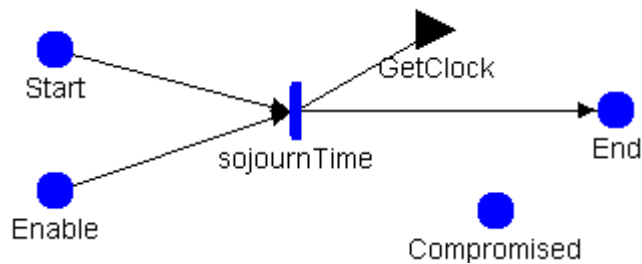
Function		#include <iostream> using namespace std; float t = BaseModelClass::LastActionTime; cout << t << endl; Enable->Mark()=1;	
Performance Variable Model: Ready2Attack			
Top Level Model Information		Child Model Name	Ready2Attack
		Model Type	SAN Model
Performance Variable : Time			
Affecting Models	Ready2Attack		
Impulse Functions			
Reward Function	(Reward is over all Available Models) if (Ready2Attack->Start->Mark() > 0) return(1); else return(0);		
Simulation	Type	Instant of Time	
	Options	Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
Parameter Statistics	Part	Stem	1000.0,2000.0,3000.0,4000.0,5000.0,6000.0,7000.0,8000.0,9000.0,10000.0,11000.0,12000.0,13000.0,14000.0,15000.0,16000.0,17000.0,18000.0,19000.0,20000.0,21000.0,22000.0,23000.0,24000.0,25000.0,26000.0,27000.0,28000.0,29000.0,30000.0,31000.0,32000.0,33000.0,34000.0,35000.0,36000.0,37000.0,38000.0,39000.0,40000.0,41000.0,42000.0,43000.0,44000.0,45000.0,46000.0,47000.0,48000.0,49000.0,50000.0,51000.0,52000.0,53000.0,54000.0,55000.0,56000.0,57000.0,58000.0,59000.0,60000.0,61000.0,62000.0,63000.0,64000.0,65000.0,66000.0,67000.0,68000.0,69000.0,70000.0,71000.0,72000.0,73000.0,74000.0,75000.0,76000.0,77000.0,78000.0,79000.0,80000.0,81000.0,82000.0,83000.0,84000.0,85000.0,86000.0,87000.0,88000.0,89000.0,90000.0,91000.0,92000.0,93000.0,94000.0,95000.0,96000.0,97000.0,98000.0,99000.0,100000.0,
Confidence	Confidence	0.95	

nc	nc	
e	e	
	Le	
	ve	
	l	
	Co	
	nfi	
	de	
	nc	
	e	0.1
	Int	
	er	
	va	
	l	

Set Study: [Ready2Attack:](#)

Experiment	Variable	Type	Value
Experiment 1	attackRate	double	0.1

Model: sojournTime (Atomic Model)



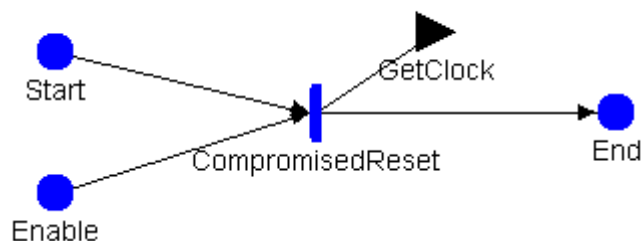
Place Attributes:	
Place Names	Initial Markings
Compromised	1
Enable	1
End	0
Start	10000
Timed Activity:	sojournTime
Distribution Parameters	Value static UserDistributions dist(0,31415); double lambda = (OH2OKRate + MH2CHRate * pow(OH2CH_scale,Compromised->Mark() && enableOLH) + MH2AccRate * pow(DelayScale,Compromised->Mark() && enableOLH)); double delay = dist.Exponential(lambda); return(delay);
Activation Predicate	(none)
Reactivation Predicate	(none)
Output Gate:	GetClock
Function	#include <iostream> using namespace std; float t = BaseModelClass::LastActionTime; cout << t << endl;

		Enable->Mark()=1;					
Performance Variable Model: sojournTime							
Top Level Model Information		Child Model Name		sojournTime			
		Model Type		SAN Model			
Performance Variable : Time							
Affecting Models		sojournTime					
Impulse Functions							
Reward Function		<i>(Reward is over all Available Models)</i> if (sojournTime->Start->Mark() > 0) return(1); else return(0);					
Simulator Statistics		Type	Instant of Time				
		Options	Include Lower Bound on Interval Estimate				
			Include Upper Bound on Interval Estimate				
			Estimate out of Range Probabilities				
		Confidence Level is Relative					
		Parameters	Start Time	1000.0,			
		Confidence	Confidence Level	0.95			
Confidence Interval	0.1						
Range Study Variable Assignments for Study <i>sojournTime</i> in Project <i>T_Exponential2</i> :							
Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
DelayScale	double	Fixed	1000.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
enableOLH	short	Manual	[0, 1]	-	-	-	-

HTML generated by Möbius using the “Document” feature.

13.8 Lomax Time Intervals Model

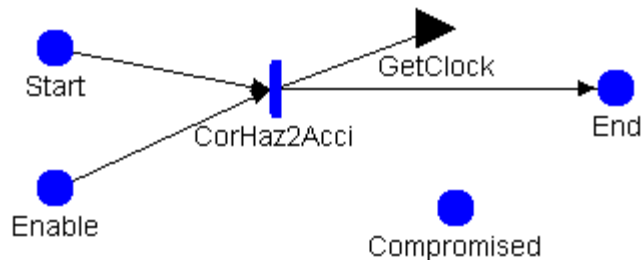
Model: CompromisedReset (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
Enable	1
End	0

Start		10000	
Timed Activity:		CompromisedReset	
Distribution Parameters		Value static UserDistributions dist(0,31415); double lambda = attackReset; double alpha = 2.0; double xm = (alpha - 1) / lambda; double delay = dist.Pareto(xm,alpha) - xm; return(delay);	
Activation Predicate		(none)	
Reactivation Predicate		(none)	
Output Gate:		GetClock	
Function		#include <iostream> using namespace std; float t = BaseModelClass::LastActionTime; cout << t << endl; Enable->Mark()=1;	
Performance Variable Model: CompromisedReset			
Top Level Model Information		Child Model Name	CompromisedReset
		Model Type	SAN Model
Performance Variable : Time			
Affecting Models	CompromisedReset		
Impulse Functions			
Reward Function	(Reward is over all Available Models) if (CompromisedReset->Start->Mark() > 0) return(1); else return(0);		
Simulator Statistics	Type	Instant of Time	
	Options	Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	1000.0,2000.0,3000.0,4000.0,5000.0,6000.0,7000.0,8000.0,9000.0,10000.0,
Confidence	Confidence Level	0.95	
	Confidence Interval	0.1	
Set Study: CompromisedReset:			
Experiment	Variable	Type	Value
Experiment 1	attackReset	double	2.0

Model: CorHaz2Acci (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
Compromised	0
Enable	1
End	0
Start	10000
Timed Activity:	CorHaz2Acci
Distribution Parameters	Value <pre> static UserDistributions dist(0,31415); double lambda = ((CH2AccRate * pow(AccidentScale,Compromised->Mark() && enableSM))*(1 - isRSSperfect) + CH2OKRate); double alpha = 2.0; double xm = (alpha - 1) / lambda; double delay = dist.Pareto(xm,alpha) - xm; return(delay); </pre>
Activation Predicate	(none)
Reactivation Predicate	(none)
Output Gate:	GetClock
Function	<pre> #include <iostream> using namespace std; float t = BaseModelClass::LastActionTime; cout << t << endl; Enable->Mark()=1; </pre>

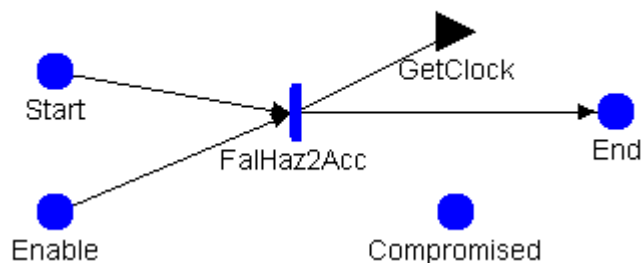
Performance Variable Model: CorHaz2Acci		
Top Level Model Information	Child Model Name	CorHaz2Acci
	Model Type	SAN Model

Performance Variable : Time		
Affecting Models	CorHaz2Acci	
Impulse Functions		
Reward Function	<i>(Reward is over all Available Models)</i> <pre> if (CorHaz2Acci->Start->Mark() > 0) return(1); else return(0); </pre>	
Simulator Statistics	Type	Instant of Time
	Options	Include Lower Bound on Interval Estimate Include Upper Bound on Interval Estimate

		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	1000.0,
	Confidence	Confidence Level	0.95
		Confidence Interval	0.1

Range Study Variable Assignments for Study <i>CorHaz2Acci</i> in Project <i>T_Lomax</i> :							
Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
enableSM	short	Manual	[0, 1]	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-

Model: FalHaz2Acc (Atomic Model)

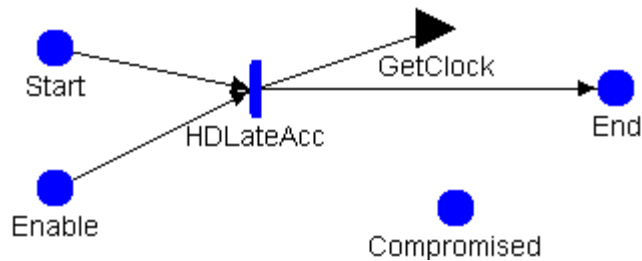


Place Attributes:	
Place Names	Initial Markings
Compromised	0
Enable	1
End	0
Start	10000
Timed Activity:	FalHaz2Acc
Distribution Parameters	Value static UserDistributions dist(0,31415); double lambda = (FH2AccRate + FH2OKRate); double alpha = 2.0; double xm = (alpha - 1) / lambda; double delay = dist.Pareto(xm,alpha) - xm; return(delay);
Activation Predicate	(none)
Reactivation Predicate	(none)
Output Gate:	GetClock
Function	#include <iostream> using namespace std; float t = BaseModelClass::LastActionTime; cout << t << endl; Enable->Mark()=1;

Performance Variable Model: FalHaz2Acc

Top Level Model Information	Child Model Name	FalHaz2Acc	
	Model Type	SAN Model	
Performance Variable : Time			
Affecting Models	FalHaz2Acc		
Impulse Functions			
Reward Function	(Reward is over all Available Models) if (FalHaz2Acc->Start->Mark() > 0) return(1); else return(0);		
Simulator Statistics	Type	Instant of Time	
	Options	Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	1000.0,
	Confidence	Confidence Level	0.95
Confidence Interval		0.1	
Set Study: FalHaz2Acc:			
Experiment	Variable	Type	Value
Experiment 1	FH2AccRate	double	1.0E-5
	FH2OKRate	double	856.3

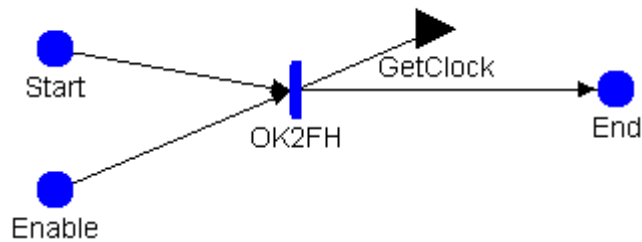
Model: HDLateAcc (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
Compromised	0
Enable	1
End	0
Start	10000
Timed Activity:	HDLateAcc
Distribution Parameters	Value static UserDistributions dist(0,31415); double lambda = ((CH2AccRate * pow(LateCH2AccScale,Compromised->Mark() && enableSM))*(1 - isRSSperfect) + CH2OKRate); double alpha = 2.0; double xm = (alpha - 1) / lambda; double delay = dist.Pareto(xm,alpha) - xm; return(delay);

Activation Predicate	(none)		
Reactivation Predicate	(none)		
Output Gate:	GetClock		
Function	#include <iostream> using namespace std; float t = BaseModelClass::LastActionTime; cout << t << endl; Enable->Mark()=1;		
Performance Variable Model: HDLateAcc			
Top Level Model Information	Child Model Name	HDLateAcc	
	Model Type	SAN Model	
Performance Variable : Time			
Affecting Models	HDLateAcc		
Impulse Functions			
Reward Function	(Reward is over all Available Models) if (HDLateAcc->Start->Mark() > 0) return(1); else return(0);		
Simulator Statistics	Type	Instant of Time	
	Options	Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	1000.0,
	Confidence	Confidence Level	0.95
Confidence Interval		0.1	
Set Study: GetClock:			
Experiment	Variable	Type	Value
Experiment 1	CH2AccRate	double	1.0E-4
	CH2OKRate	double	856.3
	LateCH2AccScale	double	10.0
	enableSM	short	0
	isRSSperfect	int	0
Experiment 2	CH2AccRate	double	1.0E-4
	CH2OKRate	double	856.3
	LateCH2AccScale	double	10.0
	enableSM	short	1
	isRSSperfect	int	0

Model: OK2FH (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
Enable	1
End	0
Start	10000
Timed Activity:	OK2FH
Distribution Parameters	Value static UserDistributions dist(0,31415); double lambda = (HazardRate * falseHazardProb); double alpha = 2.0; double xm = (alpha - 1) / lambda; double delay = dist.Pareto(xm,alpha) - xm; return(delay);
Activation Predicate	(none)
Reactivation Predicate	(none)
Output Gate:	GetClock
Function	#include <iostream> using namespace std; float t = BaseModelClass::LastActionTime; cout << t << endl; Enable->Mark()=1;

Performance Variable Model: OK2FH		
Top Level Model Information	Child Model Name	OK2FH
	Model Type	SAN Model

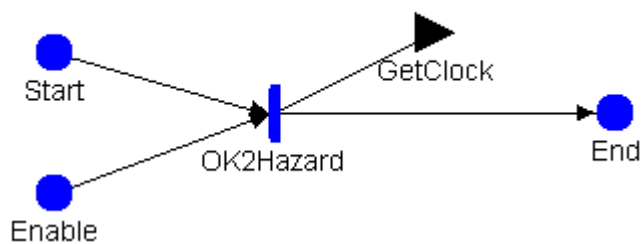
Performance Variable : Time	
A ff ec ti ng M od el s	OK2FH
I m pu ls e F	

un cti on s	
R e w a r d F u n c t i o n	(Reward is over all Available Models) <pre> if (OK2FH->Start->Mark() > 0) return(1); else return(0); </pre>
Si m u l a t o r S t a t i s t i c s	Typ Instant of Time
	O Include Lower Bound on Interval Estimate
	p Include Upper Bound on Interval Estimate
	on Estimate out of Range Probabilities
	s Confidence Level is Relative
	Pa St ra art ti m ete rs e 1000.0,2000.0,3000.0,4000.0,5000.0,6000.0,7000.0,8000.0,9000.0,10000.0,11000.0,12000.0,13000.0,14000.0,15000.0,16000.0,17000.0,18000.0,19000.0,20000.0,21000.0,22000.0,23000.0,24000.0,25000.0,26000.0,27000.0,28000.0,29000.0,30000.0,31000.0,32000.0,33000.0,34000.0,35000.0,36000.0,37000.0,38000.0,39000.0,40000.0,41000.0,42000.0,43000.0,44000.0,45000.0,46000.0,47000.0,48000.0,49000.0,50000.0,51000.0,52000.0,53000.0,54000.0,55000.0,56000.0,57000.0,58000.0,59000.0,60000.0,61000.0,62000.0,63000.0,64000.0,65000.0,66000.0,67000.0,68000.0,69000.0,70000.0,71000.0,72000.0,73000.0,74000.0,75000.0,76000.0,77000.0,78000.0,79000.0,80000.0,81000.0,82000.0,83000.0,84000.0,85000.0,86000.0,87000.0,88000.0,89000.0,90000.0,91000.0,92000.0,93000.0,94000.0,95000.0,96000.0,97000.0,98000.0,99000.0,100000.0,101000.0,102000.0,103000.0,104000.0,105000.0,106000.0,107000.0,108000.0,109000.0,110000.0,111000.0,112000.0,113000.0,114000.0,115000.0,116000.0,117000.0,118000.0,119000.0,120000.0,121000.0,122000.0,123000.0,124000.0,125000.0,126000.0,127000.0,128000.0,129000.0,130000.0,131000.0,132000.0,133000.0,134000.0,135000.0,136000.0,137000.0,138000.0,139000.0,140000.0,141000.0,142000.0,143000.0,144000.0,145000.0,146000.0,147000.0,148000.0,149000.0,150000.0,151000.0,152000.0,153000.0,154000.0,155000.0,156000.0,157000.0,158000.0,159000.0,160000.0,161000.0,162000.0,163000.0,164000.0,165000.0,166000.0,167000.0,168000.0,169000.0,170000.0,171000.0,172000.0,173000.0,174000.0,175000.0,176000.0,177000.0,178000.0,179000.0,180000.0,181000.0,182000.0,183000.0,184000.0,185000.0,186000.0,187000.0,188000.0,189000.0,190000.0,191000.0,192000.0,193000.0,194000.0,195000.0,196000.0,197000.0,198000.0,199000.0,200000.0,201000.0,202000.0,203000.0,204000.0,205000.0,206000.0,207000.0,208000.0,209000.0,210000.0,211000.0,212000.0,213000.0,214000.0,215000.0,216000.0,217000.0,218000.0,219000.0,220000.0,221000.0,222000.0,223000.0,224000.0,225000.0,226000.0,227000.0,228000.0,229000.0,230000.0,231000.0,232000.0,233000.0,234000.0,235000.0,236000.0,237000.0,238000.0,239000.0,240000.0,241000.0,242000.0,243000.0,244000.0,245000.0,246000.0,247000.0,248000.0,249000.0,250000.0,251000.0,252000.0,253000.0,254000.0,255000.0,256000.0,257000.0,258000.0,259000.0,260000.0,261000.0,262000.0,263000.0,264000.0,265000.0,266000.0,267000.0,268000.0,269000.0,270000.0,271000.0,272000.0,273000.0,274000.0,275000.0,276000.0,277000.0,278000.0,279000.0,280000.0,281000.0,282000.0,283000.0,284000.0,285000.0,286000.0,287000.0,288000.0,289000.0,290000.0,291000.0,292000.0,293000.0,294000.0,295000.0,296000.0,297000.0,298000.0,299000.0,300000.0,301000.0,302000.0,303000.0,304000.0,305000.0,306000.0,307000.0,308000.0,309000.0,310000.0,311000.0,312000.0,313000.0,314000.0,315000.0,316000.0,317000.0,318000.0,319000.0,320000.0,321000.0,322000.0,323000.0,324000.0,325000.0,326000.0,327000.0,328000.0,329000.0,330000.0,331000.0,332000.0,333000.0,334000.0,335000.0,336000.0,337000.0,338000.0,339000.0,340000.0,341000.0,342000.0,343000.0,344000.0,345000.0,346000.0,347000.0,348000.0,349000.0,350000.0,351000.0,352000.0,353000.0,354000.0,355000.0,356000.0,357000.0,358000.0,359000.0,360000.0,361000.0,362000.0,363000.0,364000.0,365000.0,366000.0,367000.0,368000.0,369000.0,370000.0,371000.0,372000.0,373000.0,374000.0,375000.0,376000.0,377000.0,378000.0,379000.0,380000.0,381000.0,382000.0,383000.0,384000.0,385000.0,386000.0,387000.0,388000.0,389000.0,390000.0,391000.0,392000.0,393000.0,394000.0,395000.0,396000.0,397000.0,398000.0,399000.0,400000.0,401000.0,402000.0,403000.0,404000.0,

Co nfi de nc e	Co nfi de nc e Le ve l	0.95
	Co nfi de nc e Int er va l	0.1

Set Study: OK2FH :			
Experiment	Variable	Type	Value
Experiment 1	HazardRate	Double	197.4
	falseHazardProb	Double	1.0E-4

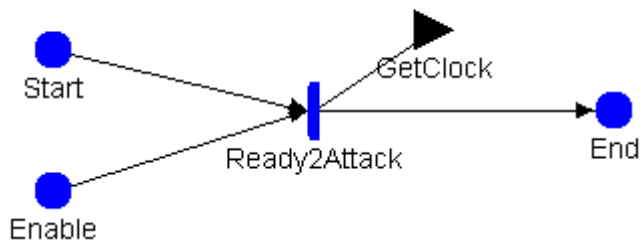
Model: OK2Hazard (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
Enable	1
End	0
Start	10000
Timed Activity:	OK2Hazard
Distribution Parameters	Value static UserDistributions dist(0,31415); double lambda = (HazardRate * (1.0 - falseHazardProb)); double alpha = 2.0; double xm = (alpha - 1) / lambda; double delay = dist.Pareto(xm,alpha) - xm; return(delay);
Activation Predicate	(none)
Reactivation Predicate	(none)
Output Gate:	GetClock
Function	#include <iostream> using namespace std; float t = BaseModelClass::LastActionTime;

	cout << t << endl; Enable->Mark()=1;		
Performance Variable Model: OK2Hazard			
Top Level Model Information	Child Model Name	OK2Hazard	
	Model Type	SAN Model	
Performance Variable : Time			
Affecting Models	OK2Hazard		
Impulse Functions			
Reward Function	(Reward is over all Available Models) if (OK2Hazard->Start->Mark() > 0) return(1); else return(0);		
Simulator Statistics	Type	Instant of Time	
	Options	Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	1000.0,
	Confidence	Confidence Level	0.95
		Confidence Interval	0.1
Set Study: OK2Hazard:			
Experiment	Variable	Type	Value
Experiment 1	HazardRate	double	197.4
	falseHazardProb	double	1.0E-4

Model: Ready2Attack (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
Enable	1
End	0
Start	10000
Timed Activity:	Ready2Attack
Distribution Parameters	Value static UserDistributions dist(0,31415); double lambda = attackRate; double alpha = 2.0; double xm = (alpha - 1) / lambda; double delay = dist.Pareto(xm,alpha) - xm;

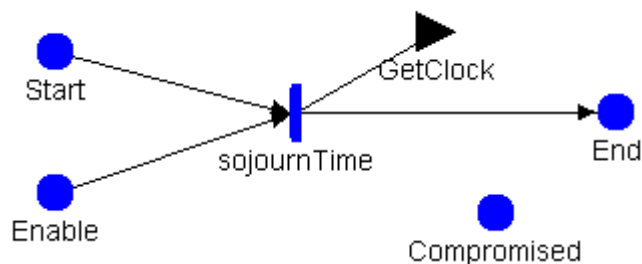
		return(delay);
Activation Predicate		(none)
Reactivation Predicate		(none)
Output Gate:		GetClock
Function		<pre>#include <iostream> using namespace std; float t = BaseModelClass::LastActionTime; cout << t << endl; Enable->Mark()=1;</pre>
Performance Variable Model: Ready2Attack		
Top Level Model Information	Child Model Name	Ready2Attack
	Model Type	SAN Model
Performance Variable : Time		
A ff ec ti ng M od el s	Ready2Attack	
I m pu ls e F un cti on s		
R e w ar d F un cti on	<i>(Reward is over all Available Models)</i> <pre>if (Ready2Attack->Start->Mark() > 0) return(1); else return(0);</pre>	
Si m u l at or St ati sti cs	Typ	Instant of Time
	O	Include Lower Bound on Interval Estimate
	pti	Include Upper Bound on Interval Estimate
	on	Estimate out of Range Probabilities
	s	Confidence Level is Relative
	Pa ra m ete rs	1000.0,2000.0,3000.0,4000.0,5000.0,6000.0,7000.0,8000.0,9000.0,10000.0,11000.0,12000.0,13000.0,14000.0,15000.0,16000.0,17000.0,18000.0,19000.0,20000.0,21000.0,22000.0,23000.0,24000.0,25000.0,26000.0,27000.0,28000.0,29000.0,30000.0,31000.0,32000.0,33000.0,34000.0,35000.0,36000.0,37000.0,38000.0,39000.0,40000.0,41000.0,42000.0,43000.0,44000.0,45000.0,46000.0,47000.0,48000.0,49000.0,50000.0,51000.0,52000.0,53000.0,54000.0,55000.0,56000.0,57000.0,58000.0,59000.0,60000.0,61000.0,62000.0,63000.0,64000.0,65000.0,66000.0,67000.0,68000.0,69000.0,70000.0,71000.0,72000.0,73000.0,74000.0,75000.0,76000.0,77000.0,78000.0,79000.0,80000.0,81000.0,82000.0

		0,83000.0,84000.0,85000.0,86000.0,87000.0,88000.0,89000.0,90000.0,91000.0,92000.0,93000.0,94000.0,95000.0,96000.0,97000.0,98000.0,99000.0,100000.0,
Confidence Level	0.95	
Confidence Interval	0.1	

Set Study: [Ready2Attack:](#)

Experiment	Variable	Type	Value
Experiment 1	attackRate	double	0.1

Model: sojournTime (Atomic Model)



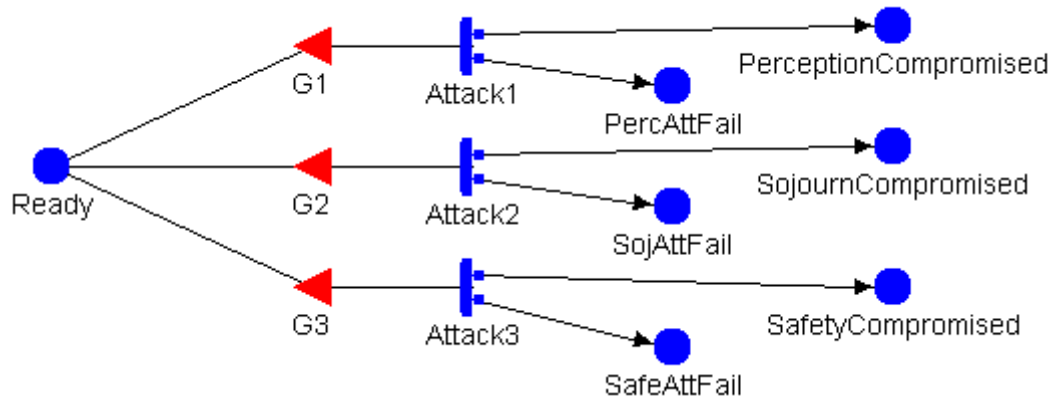
Place Attributes:	
Place Names	Initial Markings
Compromised	1
Enable	1
End	0
Start	10000
Timed Activity:	sojournTime
Distribution Parameters	Value <pre> static UserDistributions dist(0,31415); double lambda = (OH2OKRate + MH2CHRate * pow(OH2CH_scale,Compromised->Mark() && enableOLH) + MH2AccRate * pow(DelayScale,Compromised->Mark() && enableOLH)); double alpha = 2.0; double xm = (alpha - 1) / lambda; double delay = dist.Pareto(xm,alpha) - xm; return(delay); </pre>
Activation Predicate	(none)

Reactivation Predicate	(none)						
Output Gate:		GetClock					
Function	#include <iostream> using namespace std; float t = BaseModelClass::LastActionTime; cout << t << endl; Enable->Mark()=1;						
Performance Variable Model: sojournTime							
Top Level Model Information		Child Model Name	sojournTime				
		Model Type	SAN Model				
Performance Variable : Time							
Affecting Models	sojournTime						
Impulse Functions							
Reward Function	(Reward is over all Available Models) if (sojournTime->Start->Mark() > 0) return(1); else return(0);						
Simulator Statistics	Type	Instant of Time					
	Options	Include Lower Bound on Interval Estimate					
		Include Upper Bound on Interval Estimate					
		Estimate out of Range Probabilities					
		Confidence Level is Relative					
	Parameters	Start Time	1000.0,				
	Confidence	Confidence Level	0.95				
Confidence Interval		0.1					
Range Study Variable Assignments for Study sojournTime in Project T_Lomax :							
Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
DelayScale	double	Fixed	1000.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
enableOLH	short	Manual	[0, 1]	-	-	-	-

HTML generated by Möbius using the “Document” feature.

13.9 Intrusion Tolerance Models

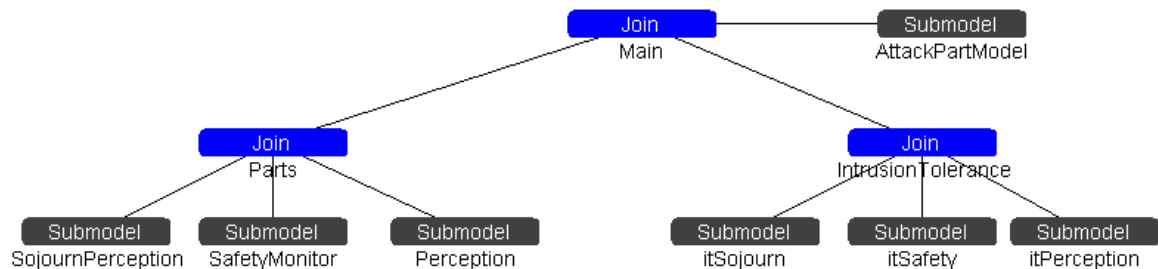
Model: AttackPartModel (Atomic Model)



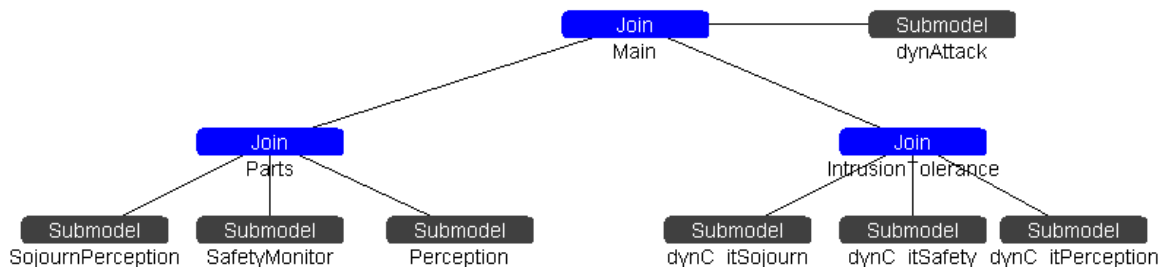
Place Attributes:	
Place Names	Initial Markings
PercAttFail	0
PerceptionCompromised	0
Ready	1
SafeAttFail	0
SafetyCompromised	0
SojAttFail	0
SojournCompromised	0
Timed Activity:	Attack1
Distribution Parameters	Rate return(percRate);
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 return(percSuccess); case 2 return(1.0-percSuccess);
Timed Activity:	Attack2
Distribution Parameters	Rate return(sojRate);
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 return(sojSuccess); case 2 return(1.0-sojSuccess);
Timed Activity:	Attack3

Distribution Parameters	Rate return(smRate);
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 return(smSuccess); case 2 return(1.0-smSuccess);
Input Gate:	G1
Predicate	(Ready->Mark() > 0)
Function	;
Input Gate:	G2
Predicate	(Ready->Mark() > 0)
Function	;
Input Gate:	G3
Predicate	(Ready->Mark() > 0)
Function	;

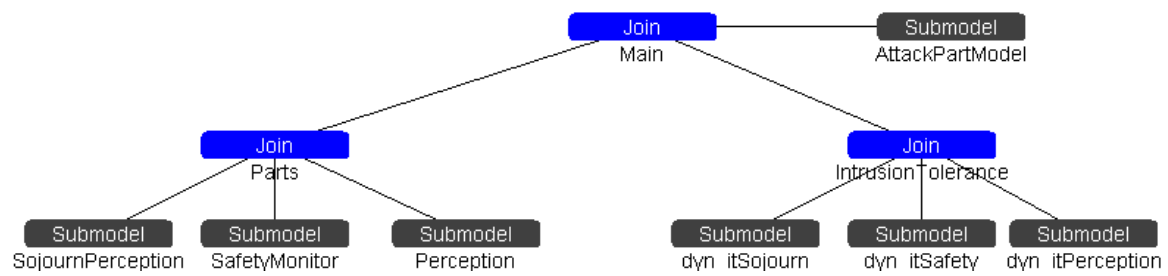
Model: ITCAVParts (Composed Model)



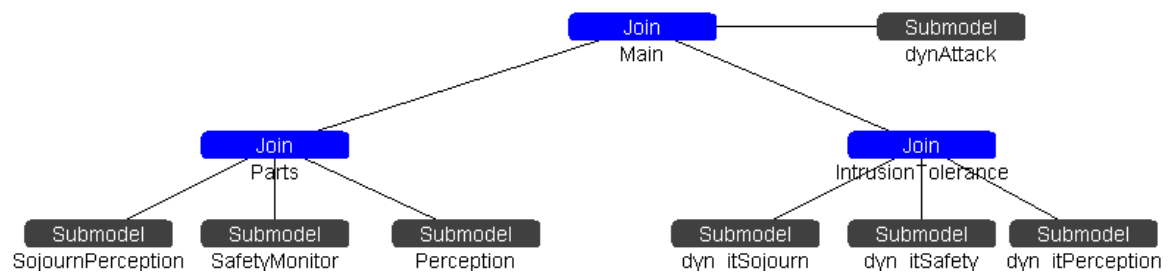
Model: ITDynAC (Composed Model)



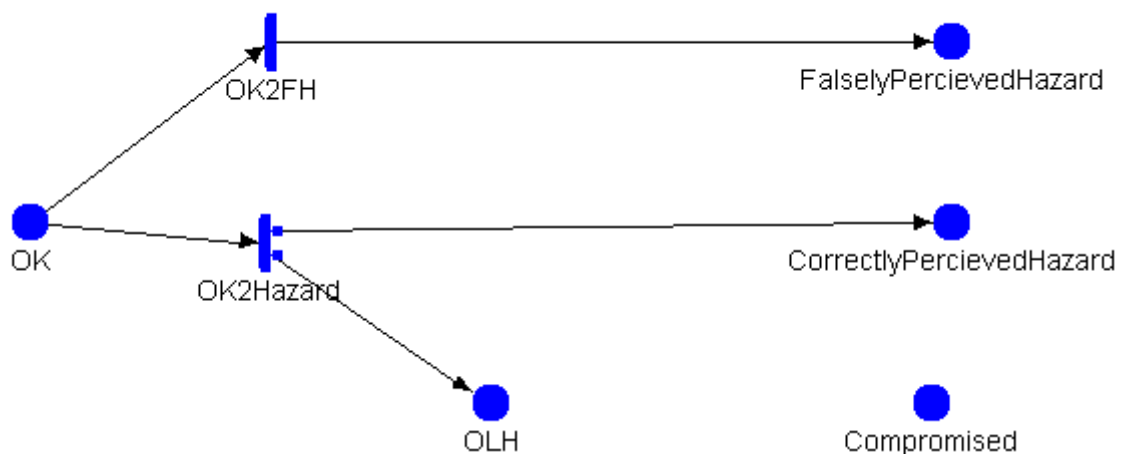
Model: ITDynFixedAttack (Composed Model)



Model: ITDynamicCAVParts (Composed Model)



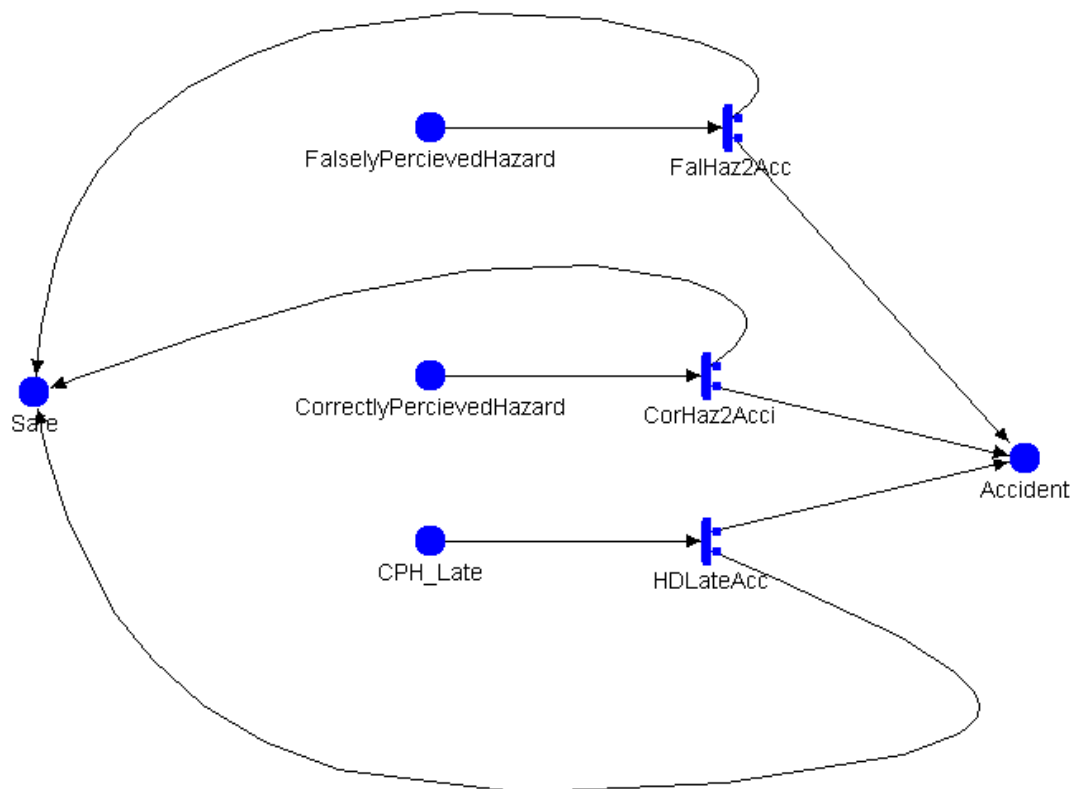
Model: Perception (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
Compromised	0
CorrectlyPercievedHazard	0
FalselyPercievedHazard	0
OK	1
OLH	0
Timed Activity:	OK2FH
Distribution Parameters	Rate return(HazardRate * falseHazardProb);
Activation Predicate	(none)
Reactivation Predicate	(none)
Timed Activity:	OK2Hazard

Distribution Parameters	Rate return(HazardRate * (1.0 - falseHazardProb));
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 return(1.0 - missHazardProb * pow(HazardScale,Compromised->Mark()) && percEnable)); case 2 return(missHazardProb * pow(HazardScale,Compromised->Mark()) && percEnable));

Model: SafetyMonitor (Atomic Model)



isRSSperfect - a boolean variable:

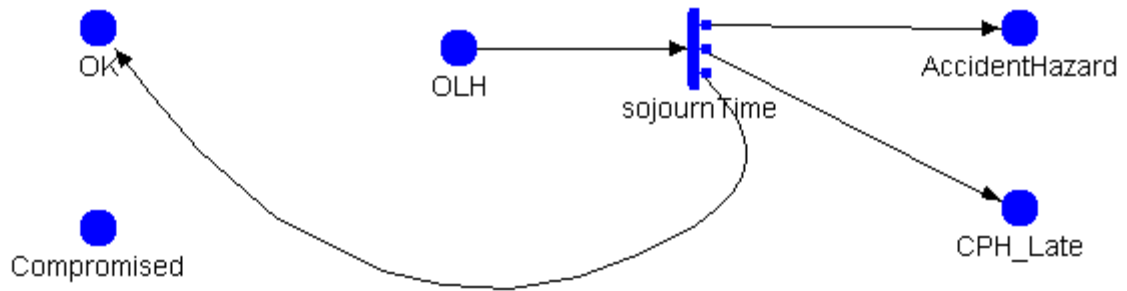
- 1: if RSS is perfect (detected hazard cannot lead to accident)
- 0: if RSS is not perfect, then the hazard can lead to an accident.

 Compromised

Place Attributes:	
Place Names	Initial Markings
Accident	0
CPH_Late	0
Compromised	0
CorrectlyPercievedHazard	0
FalselyPercievedHazard	0
Safe	1

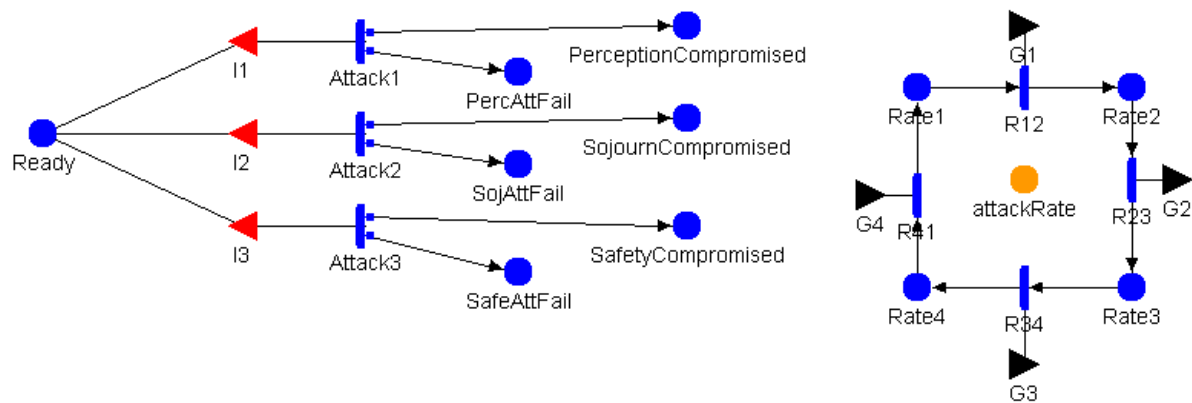
Timed Activity:	CorHaz2Acci
Distribution Parameters	Rate return((CH2AccRate * pow(AccidentScale,Compromised->Mark() && smEnable))*(1 - isRSSperfect) + CH2OKRate);
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 return(CH2OKRate/((CH2AccRate * pow(AccidentScale,Compromised->Mark() && smEnable))*(1 - isRSSperfect) + CH2OKRate)); case 2 return((CH2AccRate * pow(AccidentScale,Compromised->Mark() && smEnable))*(1 - isRSSperfect)/((CH2AccRate * pow(AccidentScale,Compromised->Mark() && smEnable))*(1 - isRSSperfect) + CH2OKRate));
Timed Activity:	FalHaz2Acc
Distribution Parameters	Rate return(FH2AccRate + FH2OKRate);
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 return(FH2OKRate / (FH2AccRate * pow(AccidentScale,Compromised->Mark() && smEnable) + FH2OKRate)); case 2 return(FH2AccRate * pow(AccidentScale,Compromised->Mark() && smEnable) / (FH2AccRate * pow(AccidentScale,Compromised->Mark() && smEnable) + FH2OKRate));
Timed Activity:	HDLateAcc
Distribution Parameters	Rate return((CH2AccRate * pow(LateCH2AccScale,Compromised->Mark() && smEnable))*(1 - isRSSperfect) + CH2OKRate);
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 return((((CH2AccRate * pow(LateCH2AccScale,Compromised->Mark() && smEnable))*(1 - isRSSperfect))/((CH2AccRate * pow(LateCH2AccScale,Compromised->Mark() && smEnable))*(1 - isRSSperfect) + CH2OKRate)); case 2 return(CH2OKRate/((CH2AccRate * pow(LateCH2AccScale,Compromised->Mark() && smEnable))*(1 - isRSSperfect) + CH2OKRate));

Model: SojournPerception (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
AccidentHazard	0
CPH_Late	0
Compromised	0
OK	1
OLH	0
Timed Activity:	sojournTime
Distribution Parameters	Rate $\text{return}(\text{OH2OKRate} + \text{MH2CHRate} * \text{pow}(\text{OH2CH_scale}, \text{Compromised} \rightarrow \text{Mark}()) \&\& \text{sojEnable}) + \text{MH2AccRate} * \text{pow}(\text{DelayScale}, \text{Compromised} \rightarrow \text{Mark}()) \&\& \text{sojEnable});$
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 $\text{return}((\text{MH2AccRate} * \text{pow}(\text{DelayScale}, \text{Compromised} \rightarrow \text{Mark}()) \&\& \text{sojEnable}) / (\text{OH2OKRate} + \text{MH2CHRate} * \text{pow}(\text{OH2CH_scale}, \text{Compromised} \rightarrow \text{Mark}()) \&\& \text{sojEnable}) + \text{MH2AccRate} * \text{pow}(\text{DelayScale}, \text{Compromised} \rightarrow \text{Mark}()) \&\& \text{sojEnable}));$ case 2 $\text{return}((\text{MH2CHRate} * \text{pow}(\text{OH2CH_scale}, \text{Compromised} \rightarrow \text{Mark}()) \&\& \text{sojEnable}) / (\text{OH2OKRate} + \text{MH2CHRate} * \text{pow}(\text{OH2CH_scale}, \text{Compromised} \rightarrow \text{Mark}()) \&\& \text{sojEnable}) + \text{MH2AccRate} * \text{pow}(\text{DelayScale}, \text{Compromised} \rightarrow \text{Mark}()) \&\& \text{sojEnable}));$ case 3 $\text{return}((\text{OH2OKRate}) / (\text{OH2OKRate} + \text{MH2CHRate} * \text{pow}(\text{OH2CH_scale}, \text{Compromised} \rightarrow \text{Mark}()) \&\& \text{sojEnable}) + \text{MH2AccRate} * \text{pow}(\text{DelayScale}, \text{Compromised} \rightarrow \text{Mark}()) \&\& \text{sojEnable}));$

Model: dynAttack (Atomic Model)

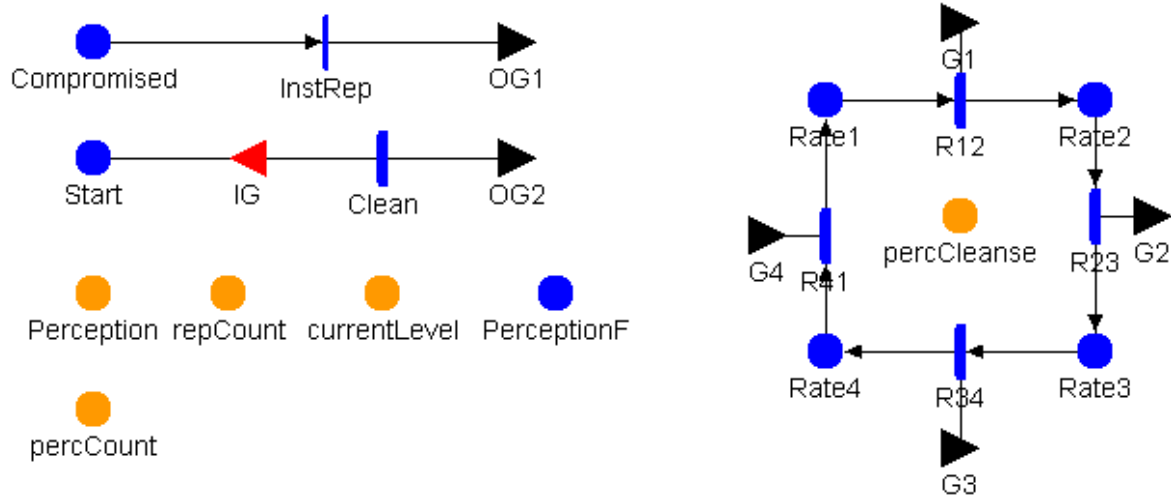


Place Attributes:	
Place Names	Initial Markings
PercAttFail	0
PerceptionCompromised	0
Rate1	1
Rate2	0
Rate3	0
Rate4	0
Ready	1
SafeAttFail	0
SafetyCompromised	0
SojAttFail	0
SojournCompromised	0
attackRate	0.01
Timed Activity:	Attack1
Distribution Parameters	Rate return(attackRate->Mark());
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 return(percSuccess); case 2 return(1.0-percSuccess);
Timed Activity:	Attack2
Distribution Parameters	Rate return(attackRate->Mark());
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 return(sojSuccess); case 2 return(1.0-sojSuccess);

Timed Activity:	Attack3
Distribution Parameters	Rate return(attackRate->Mark());
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	case 1 return(smSuccess); case 2 return(1.0-smSuccess);
Timed Activity:	R12
Distribution Parameters	Rate return(dynRate);
Activation Predicate	(none)
Reactivation Predicate	(none)
Timed Activity:	R23
Distribution Parameters	Rate return(dynRate);
Activation Predicate	(none)
Reactivation Predicate	(none)
Timed Activity:	R34
Distribution Parameters	Rate return(dynRate);
Activation Predicate	(none)
Reactivation Predicate	(none)
Timed Activity:	R41
Distribution Parameters	Rate return(dynRate);
Activation Predicate	(none)
Reactivation Predicate	(none)
Input Gate:	I1
Predicate	(Ready->Mark() > 0)
Function	;
Input Gate:	I2
Predicate	(Ready->Mark() > 0)
Function	;
Input Gate:	I3
Predicate	(Ready->Mark() > 0)
Function	;

Output Gate:	G1
Function	attackRate->Mark()=0.05;
Output Gate:	G2
Function	attackRate->Mark()=0.1;
Output Gate:	G3
Function	attackRate->Mark()=0.3;
Output Gate:	G4
Function	attackRate->Mark()=0.01;

Model: dynC_itPerception (Atomic Model)



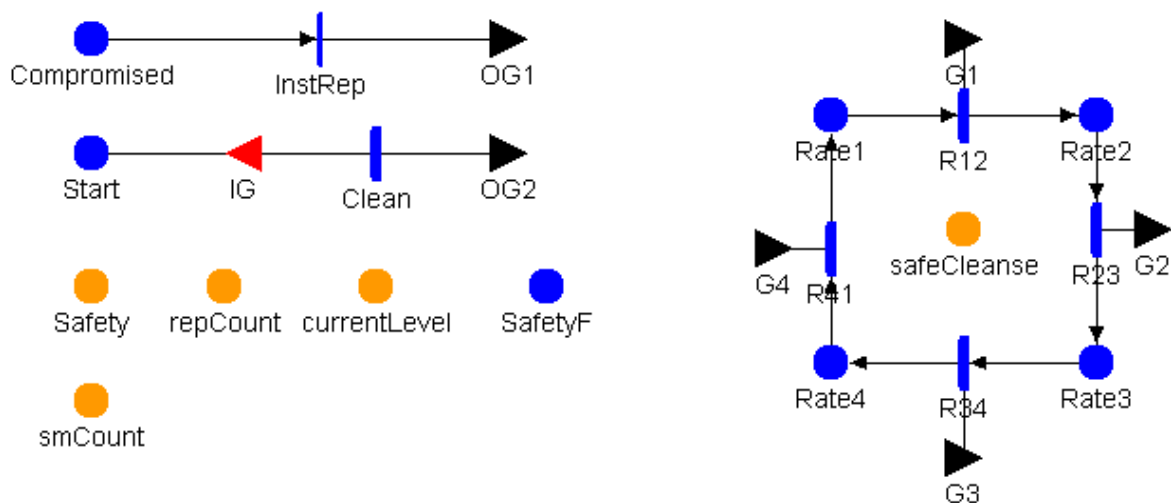
Place Attributes:	
Place Names	Initial Markings
Compromised	0
Perception	0
PerceptionF	0
Rate1	1
Rate2	0
Rate3	0
Rate4	0
Start	1
currentLevel	1
percCleanse	0.3
percCount	0
repCount	0
Timed Activity:	Clean
Distribution Parameters	Rate return(percCleanse->Mark());
Activation Predicate	(none)
Reactivation Predicate	(none)

Timed Activity:	R12
Distribution Parameters	Rate return(dynPerc);
Activation Predicate	(none)
Reactivation Predicate	(none)
Timed Activity:	R23
Distribution Parameters	Rate return(dynPerc);
Activation Predicate	(none)
Reactivation Predicate	(none)
Timed Activity:	R34
Distribution Parameters	Rate return(dynPerc);
Activation Predicate	(none)
Reactivation Predicate	(none)
Timed Activity:	R41
Distribution Parameters	Rate return(dynPerc);
Activation Predicate	(none)
Reactivation Predicate	(none)
Instantaneous Activities Without Cases:	
InstRep	
Input Gate:	IG
Predicate	(Start->Mark() > 0)
Function	;
Output Gate:	G1
Function	percCleanse->Mark()=0.5;
Output Gate:	G2
Function	percCleanse->Mark()=1.0;
Output Gate:	G3
Function	percCleanse->Mark()=2.0;
Output Gate:	G4
Function	percCleanse->Mark()=0.3;
Output Gate:	OG1
Function	<pre> int numReps = (3*currentLevel->Mark())+1; int repThreshold = (numReps + 2) / 3; for (int i=0;i<numReps;i++) { if (Perception->Index(i)->Mark()==0) </pre>

	<pre> { Perception->Index(i)->Mark()=1; break; } // END IF } // END FOR LOOP int compSum = 0; for (int i=0;i<numReps;i++) { compSum = compSum + Perception->Index(i)->Mark(); } // END FOR LOOP if (compSum >= repThreshold) { PerceptionF->Mark()=1; } else { PerceptionF->Mark()=0; } // END IF </pre>
Output Gate:	OG2
Function	<pre> int numReps = (3*currentLevel->Mark())+1; int x = repCount->Mark(); if (Perception->Index(x)->Mark() > 0) { percCount->Mark()++; } if (repCount->Mark() > numReps) { percCount->Mark()=0; } Perception->Index(x)->Mark()=0; repCount->Mark()++; if (repCount->Mark() > numReps-1) { repCount->Mark()=0; } // END IF int repThreshold = (numReps + 2) / 3; int compSum = 0; for (int i=0;i<numReps;i++) { compSum = compSum + Perception->Index(i)->Mark(); } // END FOR LOOP if (compSum >= repThreshold) { PerceptionF->Mark()=1; } else { PerceptionF->Mark()=0; } // END IF if (percCount->Mark() >= repThreshold) { </pre>

	<pre> currentLevel->Mark()++; } // END IF if (currentLevel->Mark() > percLevels) { currentLevel->Mark() = percLevels; } // END IF int prevLevel = currentLevel->Mark()--; if (currentLevel->Mark()-- < 1) { prevLevel=1; } int prevReps = (3*prevLevel)+1; int prevThreshold = (prevReps + 2) / 3; if (percCount->Mark() < prevThreshold) { currentLevel->Mark() = prevLevel; numReps = prevReps; } for (int i=0;i<numReps;i++) { if (compSum > 0) { Perception->Index(i)->Mark()=1; compSum--; } } for (int i=numReps;i<percmaxReps;i++) { Perception->Index(i)->Mark()=0; } </pre>
--	--

Model: dynC_itSafety (Atomic Model)



Place Attributes:

Place Names	Initial Markings
Compromised	0
Rate1	1
Rate2	0
Rate3	0
Rate4	0
Safety	0
SafetyF	0
Start	1
currentLevel	1
repCount	0
safeCleanse	0.3
smCount	0
Timed Activity:	Clean
Distribution Parameters	Rate return(safeCleanse->Mark());
Activation Predicate	(none)
Reactivation Predicate	(none)
Timed Activity:	R12
Distribution Parameters	Rate return(dynSafe);
Activation Predicate	(none)
Reactivation Predicate	(none)
Timed Activity:	R23
Distribution Parameters	Rate return(dynSafe);
Activation Predicate	(none)
Reactivation Predicate	(none)
Timed Activity:	R34
Distribution Parameters	Rate return(dynSafe);
Activation Predicate	(none)
Reactivation Predicate	(none)
Timed Activity:	R41
Distribution Parameters	Rate return(dynSafe);
Activation Predicate	(none)
Reactivation Predicate	(none)
Instantaneous Activities Without Cases:	
InstRep	
Input Gate:	IG
Predicate	(Start->Mark() > 0)
Function	

	;
Output Gate:	G1
Function	safeCleanse->Mark()=0.5;
Output Gate:	G2
Function	safeCleanse->Mark()=1.0;
Output Gate:	G3
Function	safeCleanse->Mark()=2.0;
Output Gate:	G4
Function	safeCleanse->Mark()=0.3;
Output Gate:	OG1
Function	<pre> int numReps = (3*currentLevel->Mark())+1; int repThreshold = (numReps + 2) / 3; for (int i=0;i<numReps;i++) { if (Safety->Index(i)->Mark()==0) { Safety->Index(i)->Mark()=1; break; } // END IF } // END FOR LOOP int compSum = 0; for (int i=0;i<numReps;i++) { compSum = compSum + Safety->Index(i)->Mark(); } // END FOR LOOP if (compSum >= repThreshold) { SafetyF->Mark()=1; } else { SafetyF->Mark()=0; } // END IF </pre>
Output Gate:	OG2
Function	<pre> int numReps = (3*currentLevel->Mark())+1; int x = repCount->Mark(); if (Safety->Index(x)->Mark() > 0) { smCount->Mark()++; } if (repCount->Mark() > numReps) { smCount->Mark()=0; } Safety->Index(x)->Mark()=0; </pre>

```

repCount->Mark()++;

if (repCount->Mark() > numReps-1)
{
    repCount->Mark()=0;
} // END IF

int repThreshold = (numReps + 2) / 3;
int compSum = 0;

for (int i=0;i<numReps;i++)
{
    compSum = compSum + Safety->Index(i)->Mark();
} // END FOR LOOP

if (compSum >= repThreshold)
{
    SafetyF->Mark()=1;
} else {
    SafetyF->Mark()=0;
} // END IF

if (smCount->Mark() >= repThreshold)
{
    currentLevel->Mark()++;
} // END IF

if (currentLevel->Mark() > safeLevels)
{
    currentLevel->Mark() = safeLevels;
} // END IF

int prevLevel = currentLevel->Mark()--;

if (currentLevel->Mark()-- < 1)
{
    prevLevel=1;
}

int prevReps = (3*prevLevel)+1;
int prevThreshold = (prevReps + 2) / 3;

if (smCount->Mark() < prevThreshold)
{
    currentLevel->Mark() = prevLevel;
    numReps = prevReps;
}

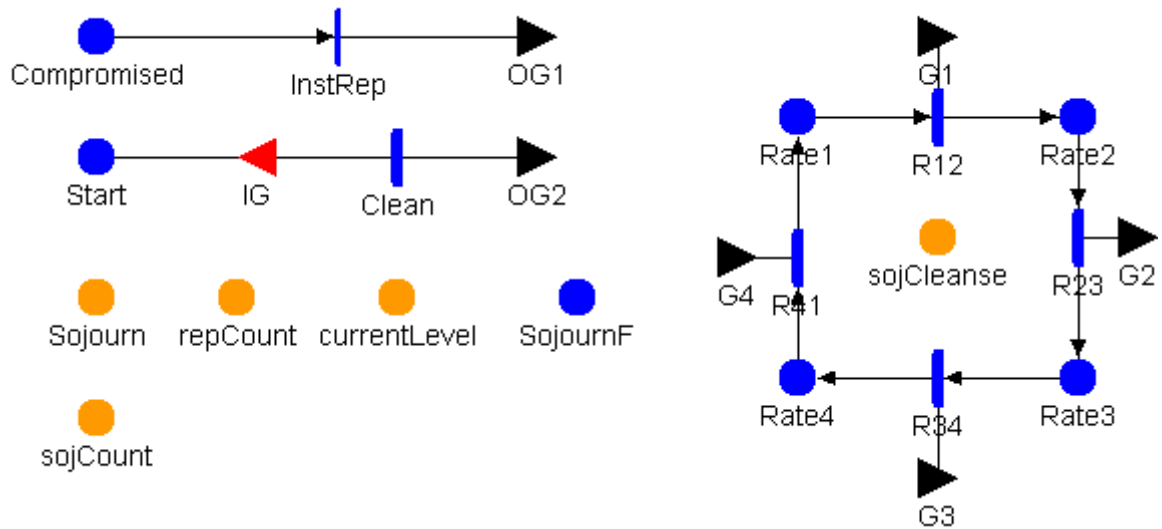
for (int i=0;i<numReps;i++)
{
    if (compSum > 0)
    {
        Safety->Index(i)->Mark()=1;
        compSum--;
    }
}

for (int i=numReps;i<safemaxReps;i++)
{
    Safety->Index(i)->Mark()=0;
}

```

	}
--	---

Model: dynC_itSojourn (Atomic Model)



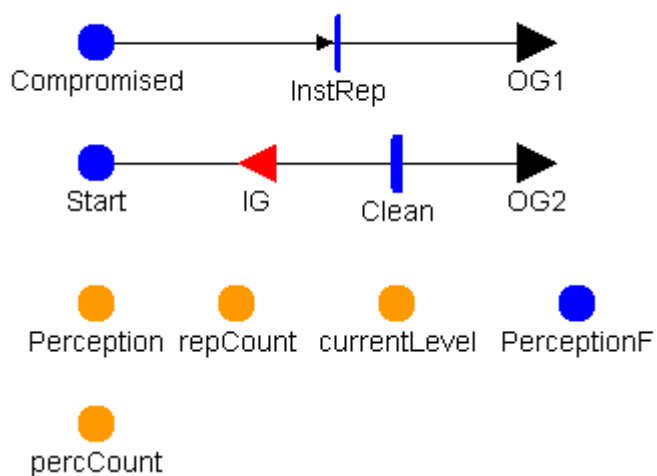
Place Attributes:	
Place Names	Initial Markings
Compromised	0
Rate1	1
Rate2	0
Rate3	0
Rate4	0
Sojourn	0
SojournF	0
Start	1
currentLevel	1
repCount	0
sojCleanse	0.3
sojCount	0
Timed Activity:	Clean
Distribution Parameters	Rate return(sojCleanse->Mark());
Activation Predicate	(none)
Reactivation Predicate	(none)
Timed Activity:	R12
Distribution Parameters	Rate return(dynSoj);
Activation Predicate	(none)
Reactivation Predicate	(none)
Timed Activity:	R23
Distribution Parameters	Rate return(dynSoj);

Activation Predicate	(none)
Reactivation Predicate	(none)
Timed Activity:	R34
Distribution Parameters	Rate return(dynSoj);
Activation Predicate	(none)
Reactivation Predicate	(none)
Timed Activity:	R41
Distribution Parameters	Rate return(dynSoj);
Activation Predicate	(none)
Reactivation Predicate	(none)
Instantaneous Activities Without Cases:	
InstRep	
Input Gate:	IG
Predicate	(Start->Mark() > 0)
Function	;
Output Gate:	G1
Function	sojCleanse->Mark()=0.5;
Output Gate:	G2
Function	sojCleanse->Mark()=1.0;
Output Gate:	G3
Function	sojCleanse->Mark()=2.0;
Output Gate:	G4
Function	sojCleanse->Mark()=0.3;
Output Gate:	OG1
Function	<pre> int numReps = (3*currentLevel->Mark())+1; int repThreshold = (numReps + 2) / 3; for (int i=0;i<numReps;i++) { if (Sojourn->Index(i)->Mark()==0) { Sojourn->Index(i)->Mark()=1; break; } // END IF } // END FOR LOOP int compSum = 0; for (int i=0;i<numReps;i++) { compSum = compSum + Sojourn->Index(i)->Mark(); </pre>

	<pre> } // END FOR LOOP if (compSum >= repThreshold) { SojournF->Mark()=1; } else { SojournF->Mark()=0; } // END IF </pre>
Output Gate:	OG2
Function	<pre> int numReps = (3*currentLevel->Mark())+1; int x = repCount->Mark(); if (Sojourn->Index(x)->Mark() > 0) { sojCount->Mark()++; } if (repCount->Mark() > numReps) { sojCount->Mark()=0; } Sojourn->Index(x)->Mark()=0; repCount->Mark()++; if (repCount->Mark() > numReps-1) { repCount->Mark()=0; } // END IF int repThreshold = (numReps + 2) / 3; int compSum = 0; for (int i=0;i<numReps;i++) { compSum = compSum + Sojourn->Index(i)->Mark(); } // END FOR LOOP if (compSum >= repThreshold) { SojournF->Mark()=1; } else { SojournF->Mark()=0; } // END IF if (sojCount->Mark() >= repThreshold) { currentLevel->Mark()++; } // END IF if (currentLevel->Mark() > sojLevels) { currentLevel->Mark() = sojLevels; } // END IF int prevLevel = currentLevel->Mark()--; if (currentLevel->Mark()-- < 1) </pre>

	<pre> { prevLevel=1; } int prevReps = (3*prevLevel)+1; int prevThreshold = (prevReps + 2) / 3; if (sojCount->Mark() < prevThreshold) { currentLevel->Mark() = prevLevel; numReps = prevReps; } for (int i=0;i<numReps;i++) { if (compSum > 0) { Sojourn->Index(i)->Mark()=1; compSum--; } } for (int i=numReps;i<sojmaxReps;i++) { Sojourn->Index(i)->Mark()=0; } </pre>
--	--

Model: dyn_itPerception (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
Compromised	0
Perception	0
PerceptionF	0
Start	1
currentLevel	1
percCount	0
repCount	0
Timed Activity:	Clean

Distribution Parameters	Rate return(percCleanse);
Activation Predicate	(none)
Reactivation Predicate	(none)
Instantaneous Activities Without Cases:	
InstRep	
Input Gate:	IG
Predicate	(Start->Mark() > 0)
Function	;
Output Gate:	OG1
Function	<pre> int numReps = (3*currentLevel->Mark())+1; int repThreshold = (numReps + 2) / 3; for (int i=0;i<numReps;i++) { if (Perception->Index(i)->Mark()==0) { Perception->Index(i)->Mark()=1; break; } // END IF } // END FOR LOOP int compSum = 0; for (int i=0;i<numReps;i++) { compSum = compSum + Perception->Index(i)->Mark(); } // END FOR LOOP if (compSum >= repThreshold) { PerceptionF->Mark()=1; } else { PerceptionF->Mark()=0; } // END IF </pre>
Output Gate:	OG2
Function	<pre> int numReps = (3*currentLevel->Mark())+1; int x = repCount->Mark(); if (Perception->Index(x)->Mark() > 0) { percCount->Mark()++; } if (repCount->Mark() > numReps) { percCount->Mark()=0; } Perception->Index(x)->Mark()=0; repCount->Mark()++; </pre>

```

if (repCount->Mark() > numReps-1)
{
    repCount->Mark()=0;
} // END IF

int repThreshold = (numReps + 2) / 3;
int compSum = 0;

for (int i=0;i<numReps;i++)
{
    compSum = compSum + Perception->Index(i)->Mark();
} // END FOR LOOP

if (compSum >= repThreshold)
{
    PerceptionF->Mark()=1;
} else {
    PerceptionF->Mark()=0;
} // END IF

if (percCount->Mark() >= repThreshold)
{
    currentLevel->Mark()++;
} // END IF

if (currentLevel->Mark() > percLevels)
{
    currentLevel->Mark() = percLevels;
} // END IF

int prevLevel = currentLevel->Mark()--;

if (currentLevel->Mark()-- < 1)
{
    prevLevel=1;
}

int prevReps = (3*prevLevel)+1;
int prevThreshold = (prevReps + 2) / 3;

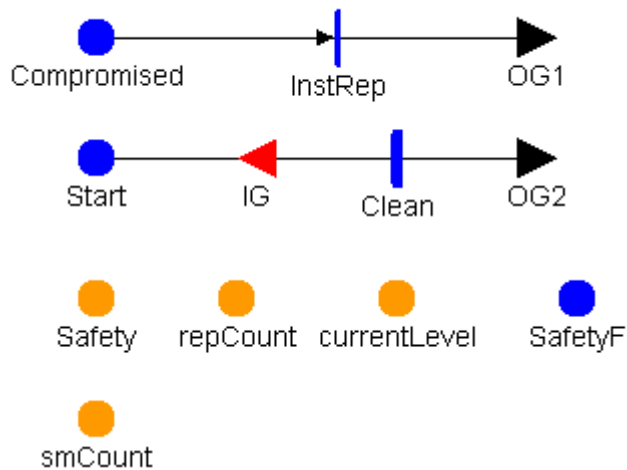
if (percCount->Mark() < prevThreshold)
{
    currentLevel->Mark() = prevLevel;
    numReps = prevReps;
}

for (int i=0;i<numReps;i++)
{
    if (compSum > 0)
    {
        Perception->Index(i)->Mark()=1;
        compSum--;
    }
}

for (int i=numReps;i<percmxReps;i++)
{
    Perception->Index(i)->Mark()=0;
}

```

Model: dyn_itSafety (Atomic Model)

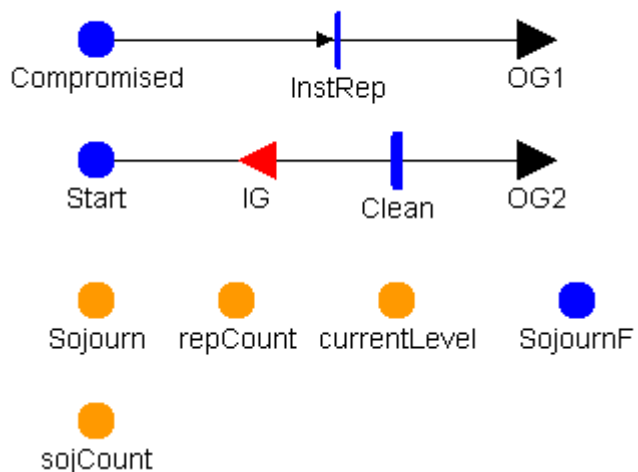


Place Attributes:	
Place Names	Initial Markings
Compromised	0
Safety	0
SafetyF	0
Start	1
currentLevel	1
repCount	0
smCount	0
Timed Activity:	Clean
Distribution Parameters	Rate return(safeCleanse);
Activation Predicate	(none)
Reactivation Predicate	(none)
Instantaneous Activities Without Cases:	
InstRep	
Input Gate:	IG
Predicate	(Start->Mark() > 0)
Function	;
Output Gate:	OG1
Function	<pre> int numReps = (3*currentLevel->Mark()+1); int repThreshold = (numReps + 2) / 3; for (int i=0;i<numReps;i++) { if (Safety->Index(i)->Mark()==0) { Safety->Index(i)->Mark()=1; break; } // END IF } // END FOR LOOP </pre>

	<pre> int compSum = 0; for (int i=0;i<numReps;i++) { compSum = compSum + Safety->Index(i)->Mark(); } // END FOR LOOP if (compSum >= repThreshold) { SafetyF->Mark()=1; } else { SafetyF->Mark()=0; } // END IF </pre>
Output Gate:	OG2
Function	<pre> int numReps = (3*currentLevel->Mark())+1; int x = repCount->Mark(); if (Safety->Index(x)->Mark() > 0) { smCount->Mark()++; } if (repCount->Mark() > numReps) { smCount->Mark()=0; } Safety->Index(x)->Mark()=0; repCount->Mark()++; if (repCount->Mark() > numReps-1) { repCount->Mark()=0; } // END IF int repThreshold = (numReps + 2) / 3; int compSum = 0; for (int i=0;i<numReps;i++) { compSum = compSum + Safety->Index(i)->Mark(); } // END FOR LOOP if (compSum >= repThreshold) { SafetyF->Mark()=1; } else { SafetyF->Mark()=0; } // END IF if (smCount->Mark() >= repThreshold) { currentLevel->Mark()++; } // END IF if (currentLevel->Mark() > safeLevels) { </pre>

	<pre> currentLevel->Mark() = safeLevels; } // END IF int prevLevel = currentLevel->Mark()-; if (currentLevel->Mark()-- < 1) { prevLevel=1; } int prevReps = (3*prevLevel)+1; int prevThreshold = (prevReps + 2) / 3; if (smCount->Mark() < prevThreshold) { currentLevel->Mark() = prevLevel; numReps = prevReps; } for (int i=0;i<numReps;i++) { if (compSum > 0) { Safety->Index(i)->Mark()=1; compSum--; } } for (int i=numReps;i<safemaxReps;i++) { Safety->Index(i)->Mark()=0; } </pre>
--	--

Model: dyn_itSojourn (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
Compromised	0
Sojourn	0
SojournF	0
Start	1

currentLevel	1
repCount	0
sojCount	0
Timed Activity:	Clean
Distribution Parameters	Rate return(sojCleanse);
Activation Predicate	(none)
Reactivation Predicate	(none)
Instantaneous Activities Without Cases:	
InstRep	
Input Gate:	IG
Predicate	(Start->Mark() > 0)
Function	;
Output Gate:	OG1
Function	<pre> int numReps = (3*currentLevel->Mark()+1; int repThreshold = (numReps + 2) / 3; for (int i=0;i<numReps;i++) { if (Sojourn->Index(i)->Mark()==0) { Sojourn->Index(i)->Mark()=1; break; } // END IF } // END FOR LOOP int compSum = 0; for (int i=0;i<numReps;i++) { compSum = compSum + Sojourn->Index(i)->Mark(); } // END FOR LOOP if (compSum >= repThreshold) { SojournF->Mark()=1; } else { SojournF->Mark()=0; } // END IF </pre>
Output Gate:	OG2
Function	<pre> int numReps = (3*currentLevel->Mark()+1; int x = repCount->Mark(); if (Sojourn->Index(x)->Mark() > 0) { sojCount->Mark()++; } if (repCount->Mark() > numReps) { </pre>

```

        sojCount->Mark()=0;
    }

    Sojourn->Index(x)->Mark()=0;
    repCount->Mark()++;

    if (repCount->Mark() > numReps-1)
    {
        repCount->Mark()=0;
    } // END IF

    int repThreshold = (numReps + 2) / 3;
    int compSum = 0;

    for (int i=0;i<numReps;i++)
    {
        compSum = compSum + Sojourn->Index(i)->Mark();
    } // END FOR LOOP

    if (compSum >= repThreshold)
    {
        SojournF->Mark()=1;
    } else {
        SojournF->Mark()=0;
    } // END IF

    if (sojCount->Mark() >= repThreshold)
    {
        currentLevel->Mark()++;
    } // END IF

    if (currentLevel->Mark() > sojLevels)
    {
        currentLevel->Mark() = sojLevels;
    } // END IF

    int prevLevel = currentLevel->Mark()-;

    if (currentLevel->Mark()- < 1)
    {
        prevLevel=1;
    }

    int prevReps = (3*prevLevel)+1;
    int prevThreshold = (prevReps + 2) / 3;

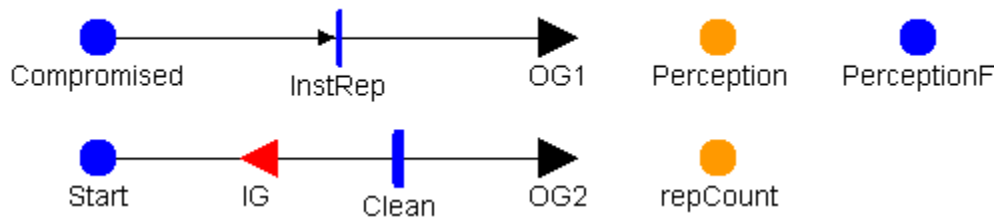
    if (sojCount->Mark() < prevThreshold)
    {
        currentLevel->Mark() = prevLevel;
        numReps = prevReps;
    }

    for (int i=0;i<numReps;i++)
    {
        if (compSum > 0)
        {
            Sojourn->Index(i)->Mark()=1;
            compSum--;
        }
    }

```

	<pre> for (int i=numReps;i<sojmaxReps;i++) { Sojourn->Index(i)->Mark()=0; } </pre>
--	---

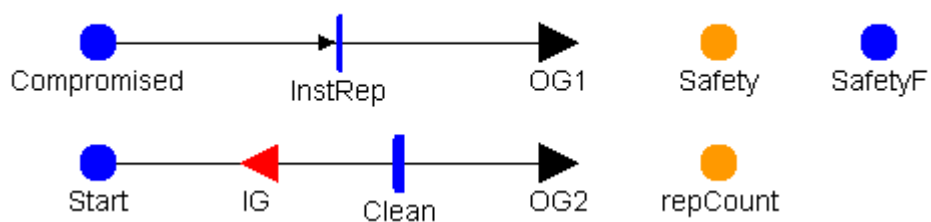
Model: itPerception (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
Compromised	0
Perception	0
PerceptionF	0
Start	1
repCount	0
Timed Activity:	Clean
Distribution Parameters	Rate return(percCleanse);
Activation Predicate	(none)
Reactivation Predicate	(none)
Instantaneous Activities Without Cases:	
InstRep	
Input Gate:	IG
Predicate	(Start->Mark() > 0)
Function	;
Output Gate:	OG1
Function	<pre> int repThreshold = (percReps + 2) / 3; for (int i=0;i<percReps;i++) { if (Perception->Index(i)->Mark()==0) { Perception->Index(i)->Mark()=1; break; } // END IF } // END FOR LOOP int compSum = 0; for (int i=0;i<percReps;i++) { compSum = compSum + Perception->Index(i)->Mark(); } </pre>

	<pre> } // END FOR LOOP if (compSum >= repThreshold) { PerceptionF->Mark()=1; } else { PerceptionF->Mark()=0; } // END IF </pre>
Output Gate:	OG2
Function	<pre> int x = repCount->Mark(); Perception->Index(x)->Mark()=0; repCount->Mark()++; if (repCount->Mark() > percReps-1) { repCount->Mark()=0; } // END IF int repThreshold = (percReps + 2) / 3; int compSum = 0; for (int i=0;i<percReps;i++) { compSum = compSum + Perception->Index(i)->Mark(); } // END FOR LOOP if (compSum >= repThreshold) { PerceptionF->Mark()=1; } else { PerceptionF->Mark()=0; } // END IF </pre>

Model: itSafety (Atomic Model)

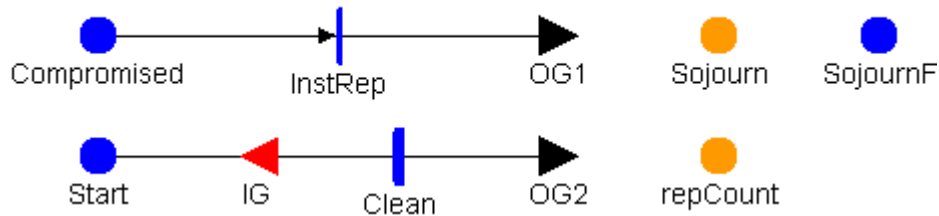


Place Attributes:	
Place Names	Initial Markings
Compromised	0
Safety	0
SafetyF	0
Start	1
repCount	0
Timed Activity:	Clean
Distribution Parameters	Rate return(smCleanse);

Activation Predicate	(none)
Reactivation Predicate	(none)
Instantaneous Activities Without Cases:	
InstRep	
Input Gate:	IG
Predicate	(Start->Mark() > 0)
Function	;
Output Gate:	OG1
Function	<pre> int repThreshold = (smReps + 2) / 3; for (int i=0;i<smReps;i++) { if (Safety->Index(i)->Mark()==0) { Safety->Index(i)->Mark()=1; break; } // END IF } // END FOR LOOP int compSum = 0; for (int i=0;i<smReps;i++) { compSum = compSum + Safety->Index(i)->Mark(); } // END FOR LOOP if (compSum >= repThreshold) { SafetyF->Mark()=1; } else { SafetyF->Mark()=0; } // END IF </pre>
Output Gate:	OG2
Function	<pre> int x = repCount->Mark(); Safety->Index(x)->Mark()=0; repCount->Mark()++; if (repCount->Mark() > smReps-1) { repCount->Mark()=0; } // END IF int repThreshold = (smReps + 2) / 3; int compSum = 0; for (int i=0;i<smReps;i++) { compSum = compSum + Safety->Index(i)->Mark(); } // END FOR LOOP if (compSum >= repThreshold) { </pre>

	<pre> SafetyF->Mark()=1; } else { SafetyF->Mark()=0; } // END IF </pre>
--	---

Model: itSojourn (Atomic Model)



Place Attributes:	
Place Names	Initial Markings
Compromised	0
Sojourn	0
SojournF	0
Start	1
repCount	0
Timed Activity:	Clean
Distribution Parameters	Rate return(sojCleanse);
Activation Predicate	(none)
Reactivation Predicate	(none)
Instantaneous Activities Without Cases:	
InstRep	
Input Gate:	IG
Predicate	(Start->Mark() > 0)
Function	;
Output Gate:	OG1
Function	<pre> int repThreshold = (sojReps + 2) / 3; for (int i=0;i<sojReps;i++) { if (Sojourn->Index(i)->Mark()==0) { Sojourn->Index(i)->Mark()=1; break; } // END IF } // END FOR LOOP int compSum = 0; for (int i=0;i<sojReps;i++) { compSum = compSum + Sojourn->Index(i)->Mark(); } // END FOR LOOP </pre>

	if (compSum >= repThreshold) { SojournF->Mark()=1; } else { SojournF->Mark()=0; } // END IF		
Output Gate:	OG2		
Function	int x = repCount->Mark(); Sojourn->Index(x)->Mark()=0; repCount->Mark()++; if (repCount->Mark() > sojReps-1) { repCount->Mark()=0; } // END IF int repThreshold = (sojReps + 2) / 3; int compSum = 0; for (int i=0;i<sojReps;i++) { compSum = compSum + Sojourn->Index(i)->Mark(); } // END FOR LOOP if (compSum >= repThreshold) { SojournF->Mark()=1; } else { SojournF->Mark()=0; } // END IF		
Performance Variable Model: reward_DynAC			
Top Level Model Information		Child Model Name	ITDynAC
		Model Type	Rep/Join
Performance Variable : TimeToAccident			
Affecting Models	SafetyMonitor		
Impulse Functions			
Reward Function	(Reward is over all Available Models) if(SafetyMonitor->Accident->Mark() > 0) return(1); else return(0);		
Simulator Statistics	Type	Instant of Time	
	Options	Estimate Mean	
		Estimate Variance	
		Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate Distribution	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	100.0,1100.0,2100.0,3100.0,4100.0,5100.0,6100.0,7100.0,8100.0,9100.0,

	Distributions	Lower Bound on Distribution Estimate	0.0	
		Upper Bound on Distribution Estimate	1.0	
		Step Size on Distribution Estimate	0.1	
	Confidence	Confidence Level	0.95	
		Confidence Interval	0.05	
Performance Variable Model: reward_DynCAVPparts				
Top Level Model Information		Child Model Name	ITDynamicCAVPparts	
		Model Type	Rep/Join	
Performance Variable : TimeToAccident				
Affecting Models	SafetyMonitor			
Impulse Functions				
Reward Function	(Reward is over all Available Models)			
	if(SafetyMonitor->Accident->Mark() > 0) return(1); else return(0);			
Simulator Statistics	Type	Instant of Time		
	Options	Estimate Mean		
		Estimate Variance		
		Include Lower Bound on Interval Estimate		
		Include Upper Bound on Interval Estimate		
		Estimate Distribution		
		Estimate out of Range Probabilities		
	Confidence Level is Relative			
	Parameters	Start Time	100.0,1100.0,2100.0,3100.0,4100.0,5100.0,6100.0,7100.0,8100.0,9100.0,	
	Distributions	Lower Bound on Distribution Estimate	0.0	
		Upper Bound on Distribution Estimate	1.0	
		Step Size on Distribution Estimate	0.1	
	Confidence	Confidence Level	0.95	
		Confidence Interval	0.05	

Performance Variable Model: reward_DynFixAttack		
Top Level Model Information	Child Model Name	ITDynFixedAttack
	Model Type	Rep/Join

Performance Variable : TimeToAccident			
Affecting Models	SafetyMonitor		
Impulse Functions			
Reward Function	<i>(Reward is over all Available Models)</i> if(SafetyMonitor->Accident->Mark() > 0) return(1); else return(0);		
Simulator Statistics	Type	Instant of Time	
	Options	Estimate Mean	
		Estimate Variance	
		Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate Distribution	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	100.0,1100.0,2100.0,3100.0,4100.0,5100.0,6100.0,7100.0,8100.0,9100.0,
	Distributions	Lower Bound on Distribution Estimate	0.0
		Upper Bound on Distribution Estimate	1.0
		Step Size on Distribution Estimate	0.1
	Confidence	Confidence Level	0.95
		Confidence Interval	0.05

Performance Variable Model: reward_ITCAVParts		
Top Level Model Information	Child Model Name	ITCAVParts
	Model Type	Rep/Join

Performance Variable : TimeToAccident			
Affecting Models	SafetyMonitor		
Impulse Functions			
Reward Function	<i>(Reward is over all Available Models)</i> if(SafetyMonitor->Accident->Mark() > 0) return(1); else return(0);		
Simulator Statistics	Type	Instant of Time	
	Options	Estimate Mean	

		Estimate Variance	
		Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate Distribution	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	100.0,1100.0,2100.0,3100.0,4100.0,5100.0,6100.0,7100.0,8100.0,9100.0,
	Distributions	Lower Bound on Distribution Estimate	0.0
		Upper Bound on Distribution Estimate	1.0
		Step Size on Distribution Estimate	0.1
	Confidence	Confidence Level	0.95
		Confidence Interval	0.05

Range Study Variable Assignments for Study *st_CAVDynFixedAttack* in Project *CAV_ITModel* :

Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-
percCleanse	double	Fixed	0.3	-	-	-	-
percEnable	short	Manual	[0, 1]	-	-	-	-
percLevels	int	Fixed	3	-	-	-	-
percRate	double	Fixed	0.1	-	-	-	-
percSuccess	double	Fixed	0.2	-	-	-	-
percmxReps	int	Fixed	100	-	-	-	-
safeCleanse	double	Fixed	0.3	-	-	-	-
safeLevels	int	Fixed	3	-	-	-	-
safemaxReps	int	Fixed	100	-	-	-	-

smEnable	short	Manual	[0, 1]	-	-	-	-
smRate	double	Fixed	0.1	-	-	-	-
smSuccess	double	Fixed	0.2	-	-	-	-
sojCleanse	double	Fixed	0.3	-	-	-	-
sojEnable	short	Manual	[0, 1]	-	-	-	-
sojLevels	int	Fixed	3	-	-	-	-
sojRate	double	Fixed	0.1	-	-	-	-
sojSuccess	double	Fixed	0.2	-	-	-	-
sojmaxReps	int	Fixed	100	-	-	-	-

Range Study Variable Assignments for Study *st_CAVPartAC* in Project *CAV_ITModel* :

Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
dynPerc	double	Fixed	0.001	-	-	-	-
dynRate	double	Fixed	0.001	-	-	-	-
dynSafe	double	Fixed	0.001	-	-	-	-
dynSoj	double	Fixed	0.001	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-
percEnable	short	Manual	[0, 1]	-	-	-	-
percLevels	int	Fixed	3	-	-	-	-
percSuccess	double	Fixed	0.2	-	-	-	-
percmaxReps	int	Fixed	100	-	-	-	-
safeLevels	int	Fixed	3	-	-	-	-
safemaxReps	int	Fixed	100	-	-	-	-
smEnable	short	Manual	[0, 1]	-	-	-	-
smSuccess	double	Fixed	0.2	-	-	-	-
sojEnable	short	Manual	[0, 1]	-	-	-	-
sojLevels	int	Fixed	3	-	-	-	-
sojSuccess	double	Fixed	0.2	-	-	-	-
sojmaxReps	int	Fixed	100	-	-	-	-

Range Study Variable Assignments for Study *st_CAVPartDynamic* in Project *CAV_ITModel* :

Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-

CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
dynRate	double	Fixed	0.001	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-
isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-
percCleanse	double	Fixed	0.3	-	-	-	-
percEnable	short	Manual	[0, 1]	-	-	-	-
percLevels	int	Fixed	3	-	-	-	-
percSuccess	double	Fixed	0.2	-	-	-	-
percmxReps	int	Fixed	100	-	-	-	-
safeCleanse	double	Fixed	0.3	-	-	-	-
safeLevels	int	Fixed	3	-	-	-	-
safemaxReps	int	Fixed	100	-	-	-	-
smEnable	short	Manual	[0, 1]	-	-	-	-
smSuccess	double	Fixed	0.2	-	-	-	-
sojCleanse	double	Fixed	0.3	-	-	-	-
sojEnable	short	Manual	[0, 1]	-	-	-	-
sojLevels	int	Fixed	3	-	-	-	-
sojSuccess	double	Fixed	0.2	-	-	-	-
sojmaxReps	int	Fixed	100	-	-	-	-

Range Study Variable Assignments for Study *st_CAVPartR4* in Project *CAV_ITModel* :

Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
AccidentScale	double	Fixed	10.0	-	-	-	-
CH2AccRate	double	Fixed	1.0E-4	-	-	-	-
CH2OKRate	double	Fixed	856.3	-	-	-	-
DelayScale	double	Fixed	1000.0	-	-	-	-
FH2AccRate	double	Fixed	1.0E-5	-	-	-	-
FH2OKRate	double	Fixed	856.3	-	-	-	-
HazardRate	double	Fixed	197.4	-	-	-	-
HazardScale	double	Fixed	10.0	-	-	-	-
LateCH2AccScale	double	Fixed	10.0	-	-	-	-
MH2AccRate	double	Fixed	0.001	-	-	-	-
MH2CHRate	double	Fixed	2250.0	-	-	-	-
OH2CH_scale	double	Fixed	10.0	-	-	-	-
OH2OKRate	double	Fixed	1000.0	-	-	-	-
falseHazardProb	double	Fixed	1.0E-4	-	-	-	-

isRSSperfect	int	Fixed	0	-	-	-	-
missHazardProb	double	Fixed	0.05	-	-	-	-
percCleanse	double	Fixed	0.3	-	-	-	-
percEnable	short	Manual	[0, 1]	-	-	-	-
percRate	double	Fixed	0.1	-	-	-	-
percReps	int	Fixed	4	-	-	-	-
percSuccess	double	Fixed	0.2	-	-	-	-
smCleanse	double	Fixed	0.3	-	-	-	-
smEnable	short	Manual	[0, 1]	-	-	-	-
smRate	double	Fixed	0.1	-	-	-	-
smReps	int	Fixed	4	-	-	-	-
smSuccess	double	Fixed	0.2	-	-	-	-
sojCleanse	double	Fixed	0.3	-	-	-	-
sojEnable	short	Manual	[0, 1]	-	-	-	-
sojRate	double	Fixed	0.1	-	-	-	-
sojReps	int	Fixed	4	-	-	-	-
sojSuccess	double	Fixed	0.2	-	-	-	-

Set Study: <u>st_R4CleanVar</u> :			
Experiment	Variable	Type	Value
Experiment 1	AccidentScale	double	10.0
	CH2AccRate	double	1.0E-4
	CH2OKRate	double	856.3
	DelayScale	double	1000.0
	FH2AccRate	double	1.0E-5
	FH2OKRate	double	856.3
	HazardRate	double	197.4
	HazardScale	double	10.0
	LateCH2AccScale	double	10.0
	MH2AccRate	double	0.001
	MH2CHRate	double	2250.0
	OH2CH_scale	double	10.0
	OH2OKRate	double	1000.0
	falseHazardProb	double	1.0E-4
	isRSSperfect	int	0
	missHazardProb	double	0.05
	percCleanse	double	1.0
	percEnable	short	1
	percRate	double	0.1
	percReps	int	4
	percSuccess	double	0.2
	smCleanse	double	1.0
	smEnable	short	1
	smRate	double	0.1
	smReps	int	4
	smSuccess	double	0.2
	sojCleanse	double	1.0
	sojEnable	short	1

	sojRate	double	0.1
	sojReps	int	4
	sojSuccess	double	0.2
Experiment 2	AccidentScale	double	10.0
	CH2AccRate	double	1.0E-4
	CH2OKRate	double	856.3
	DelayScale	double	1000.0
	FH2AccRate	double	1.0E-5
	FH2OKRate	double	856.3
	HazardRate	double	197.4
	HazardScale	double	10.0
	LateCH2AccScale	double	10.0
	MH2AccRate	double	0.001
	MH2CHRate	double	2250.0
	OH2CH_scale	double	10.0
	OH2OKRate	double	1000.0
	falseHazardProb	double	1.0E-4
	isRSSperfect	int	0
	missHazardProb	double	0.05
	percCleanse	double	0.1
	percEnable	short	1
	percRate	double	0.1
	percReps	int	4
	percSuccess	double	0.2
	smCleanse	double	0.1
	smEnable	short	1
	smRate	double	0.1
	smReps	int	4
	smSuccess	double	0.2
	sojCleanse	double	0.1
	sojEnable	short	1
	sojRate	double	0.1
	sojReps	int	4
	sojSuccess	double	0.2
Experiment 3	AccidentScale	double	10.0
	CH2AccRate	double	1.0E-4
	CH2OKRate	double	856.3
	DelayScale	double	1000.0
	FH2AccRate	double	1.0E-5
	FH2OKRate	double	856.3
	HazardRate	double	197.4
	HazardScale	double	10.0
	LateCH2AccScale	double	10.0
	MH2AccRate	double	0.001
	MH2CHRate	double	2250.0
	OH2CH_scale	double	10.0
	OH2OKRate	double	1000.0

	falseHazardProb	double	1.0E-4
	isRSSperfect	int	0
	missHazardProb	double	0.05
	percCleanse	double	0.05
	percEnable	short	1
	percRate	double	0.1
	percReps	int	4
	percSuccess	double	0.2
	smCleanse	double	0.05
	smEnable	short	1
	smRate	double	0.1
	smReps	int	4
	smSuccess	double	0.2
	sojCleanse	double	0.05
	sojEnable	short	1
	sojRate	double	0.1
	sojReps	int	4
	sojSuccess	double	0.2
Experiment 4	AccidentScale	double	10.0
	CH2AccRate	double	1.0E-4
	CH2OKRate	double	856.3
	DelayScale	double	1000.0
	FH2AccRate	double	1.0E-5
	FH2OKRate	double	856.3
	HazardRate	double	197.4
	HazardScale	double	10.0
	LateCH2AccScale	double	10.0
	MH2AccRate	double	0.001
	MH2CHRate	double	2250.0
	OH2CH_scale	double	10.0
	OH2OKRate	double	1000.0
	falseHazardProb	double	1.0E-4
	isRSSperfect	int	0
	missHazardProb	double	0.05
	percCleanse	double	0.01
	percEnable	short	1
	percRate	double	0.1
	percReps	int	4
	percSuccess	double	0.2
	smCleanse	double	0.01
	smEnable	short	1
	smRate	double	0.1
	smReps	int	4
	smSuccess	double	0.2
	sojCleanse	double	0.01
	sojEnable	short	1
	sojRate	double	0.1

	sojReps	int	4
	sojSuccess	double	0.2
Experiment 5	AccidentScale	double	10.0
	CH2AccRate	double	1.0E-4
	CH2OKRate	double	856.3
	DelayScale	double	1000.0
	FH2AccRate	double	1.0E-5
	FH2OKRate	double	856.3
	HazardRate	double	197.4
	HazardScale	double	10.0
	LateCH2AccScale	double	10.0
	MH2AccRate	double	0.001
	MH2CHRate	double	2250.0
	OH2CH_scale	double	10.0
	OH2OKRate	double	1000.0
	falseHazardProb	double	1.0E-4
	isRSSperfect	int	0
	missHazardProb	double	0.05
	percCleanse	double	0.001
	percEnable	short	1
	percRate	double	0.1
	percReps	int	4
	percSuccess	double	0.2
	smCleanse	double	0.001
	smEnable	short	1
	smRate	double	0.1
	smReps	int	4
	smSuccess	double	0.2
	sojCleanse	double	0.001
	sojEnable	short	1
	sojRate	double	0.1
	sojReps	int	4
	sojSuccess	double	0.2
Experiment 6	AccidentScale	double	10.0
	CH2AccRate	double	1.0E-4
	CH2OKRate	double	856.3
	DelayScale	double	1000.0
	FH2AccRate	double	1.0E-5
	FH2OKRate	double	856.3
	HazardRate	double	197.4
	HazardScale	double	10.0
	LateCH2AccScale	double	10.0
	MH2AccRate	double	0.001
	MH2CHRate	double	2250.0
	OH2CH_scale	double	10.0
	OH2OKRate	double	1000.0
	falseHazardProb	double	1.0E-4

	isRSSperfect	int	0
	missHazardProb	double	0.05
	percCleanse	double	0.3
	percEnable	short	1
	percRate	double	0.1
	percReps	int	4
	percSuccess	double	0.2
	smCleanse	double	0.3
	smEnable	short	1
	smRate	double	0.1
	smReps	int	4
	smSuccess	double	0.2
	sojCleanse	double	0.3
	sojEnable	short	1
	sojRate	double	0.1
	sojReps	int	4
	sojSuccess	double	0.2
Experiment 7	AccidentScale	double	10.0
	CH2AccRate	double	1.0E-4
	CH2OKRate	double	856.3
	DelayScale	double	1000.0
	FH2AccRate	double	1.0E-5
	FH2OKRate	double	856.3
	HazardRate	double	197.4
	HazardScale	double	10.0
	LateCH2AccScale	double	10.0
	MH2AccRate	double	0.001
	MH2CHRate	double	2250.0
	OH2CH_scale	double	10.0
	OH2OKRate	double	1000.0
	falseHazardProb	double	1.0E-4
	isRSSperfect	int	0
	missHazardProb	double	0.05
	percCleanse	double	0.5
	percEnable	short	1
	percRate	double	0.1
	percReps	int	4
	percSuccess	double	0.2
	smCleanse	double	0.5
	smEnable	short	1
	smRate	double	0.1
	smReps	int	4
	smSuccess	double	0.2
	sojCleanse	double	0.5
	sojEnable	short	1
	sojRate	double	0.1
	sojReps	int	4

	sojSuccess	double	0.2
Set Study: st_RepVar:			
Experiment	Variable	Type	Value
Experiment 1	AccidentScale	double	10.0
	CH2AccRate	double	1.0E-4
	CH2OKRate	double	856.3
	DelayScale	double	1000.0
	FH2AccRate	double	1.0E-5
	FH2OKRate	double	856.3
	HazardRate	double	197.4
	HazardScale	double	10.0
	LateCH2AccScale	double	10.0
	MH2AccRate	double	0.001
	MH2CHRate	double	2250.0
	OH2CH_scale	double	10.0
	OH2OKRate	double	1000.0
	falseHazardProb	double	1.0E-4
	isRSSperfect	int	0
	missHazardProb	double	0.05
	percCleanse	double	0.3
	percEnable	short	1
	percRate	double	0.1
	percReps	int	4
	percSuccess	double	0.2
	smCleanse	double	0.3
	smEnable	short	1
	smRate	double	0.1
	smReps	int	4
	smSuccess	double	0.2
	sojCleanse	double	0.3
	sojEnable	short	1
	sojRate	double	0.1
	sojReps	int	4
	sojSuccess	double	0.2
Experiment 2	AccidentScale	double	10.0
	CH2AccRate	double	1.0E-4
	CH2OKRate	double	856.3
	DelayScale	double	1000.0
	FH2AccRate	double	1.0E-5
	FH2OKRate	double	856.3
	HazardRate	double	197.4
	HazardScale	double	10.0
	LateCH2AccScale	double	10.0
	MH2AccRate	double	0.001
	MH2CHRate	double	2250.0
	OH2CH_scale	double	10.0
	OH2OKRate	double	1000.0

	falseHazardProb	double	1.0E-4
	isRSSperfect	int	0
	missHazardProb	double	0.05
	percCleanse	double	0.3
	percEnable	short	1
	percRate	double	0.1
	percReps	int	7
	percSuccess	double	0.2
	smCleanse	double	0.3
	smEnable	short	1
	smRate	double	0.1
	smReps	int	7
	smSuccess	double	0.2
	sojCleanse	double	0.3
	sojEnable	short	1
	sojRate	double	0.1
	sojReps	int	7
	sojSuccess	double	0.2
Experiment 3	AccidentScale	double	10.0
	CH2AccRate	double	1.0E-4
	CH2OKRate	double	856.3
	DelayScale	double	1000.0
	FH2AccRate	double	1.0E-5
	FH2OKRate	double	856.3
	HazardRate	double	197.4
	HazardScale	double	10.0
	LateCH2AccScale	double	10.0
	MH2AccRate	double	0.001
	MH2CHRate	double	2250.0
	OH2CH_scale	double	10.0
	OH2OKRate	double	1000.0
	falseHazardProb	double	1.0E-4
	isRSSperfect	int	0
	missHazardProb	double	0.05
	percCleanse	double	0.3
	percEnable	short	1
	percRate	double	0.1
	percReps	int	10
	percSuccess	double	0.2
	smCleanse	double	0.3
	smEnable	short	1
	smRate	double	0.1
	smReps	int	10
	smSuccess	double	0.2
	sojCleanse	double	0.3
	sojEnable	short	1
	sojRate	double	0.1

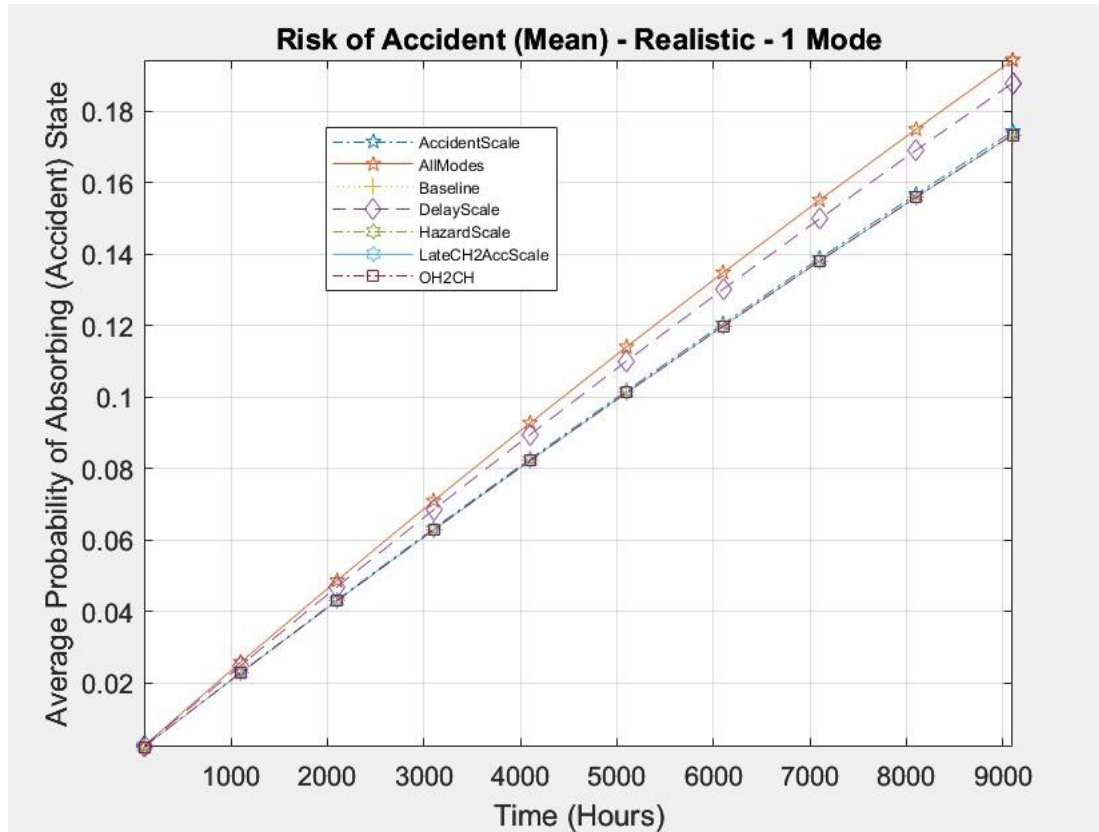
	sojReps	int	10
	sojSuccess	double	0.2

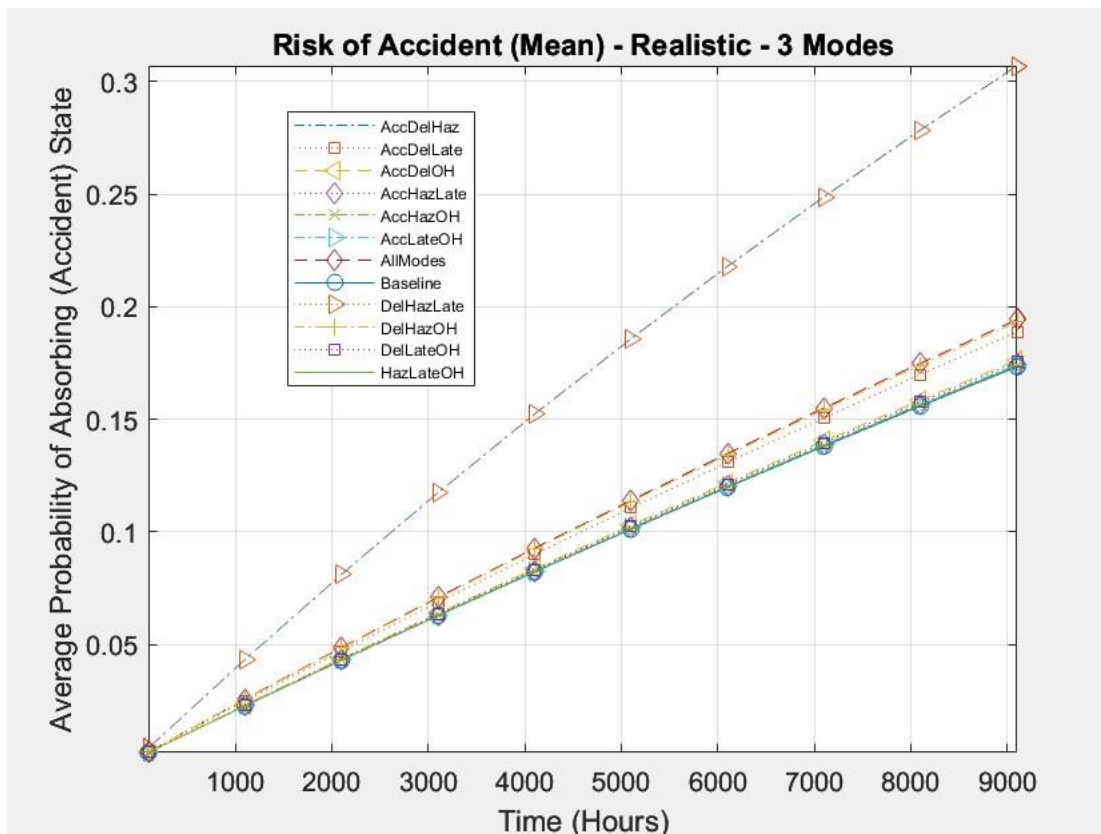
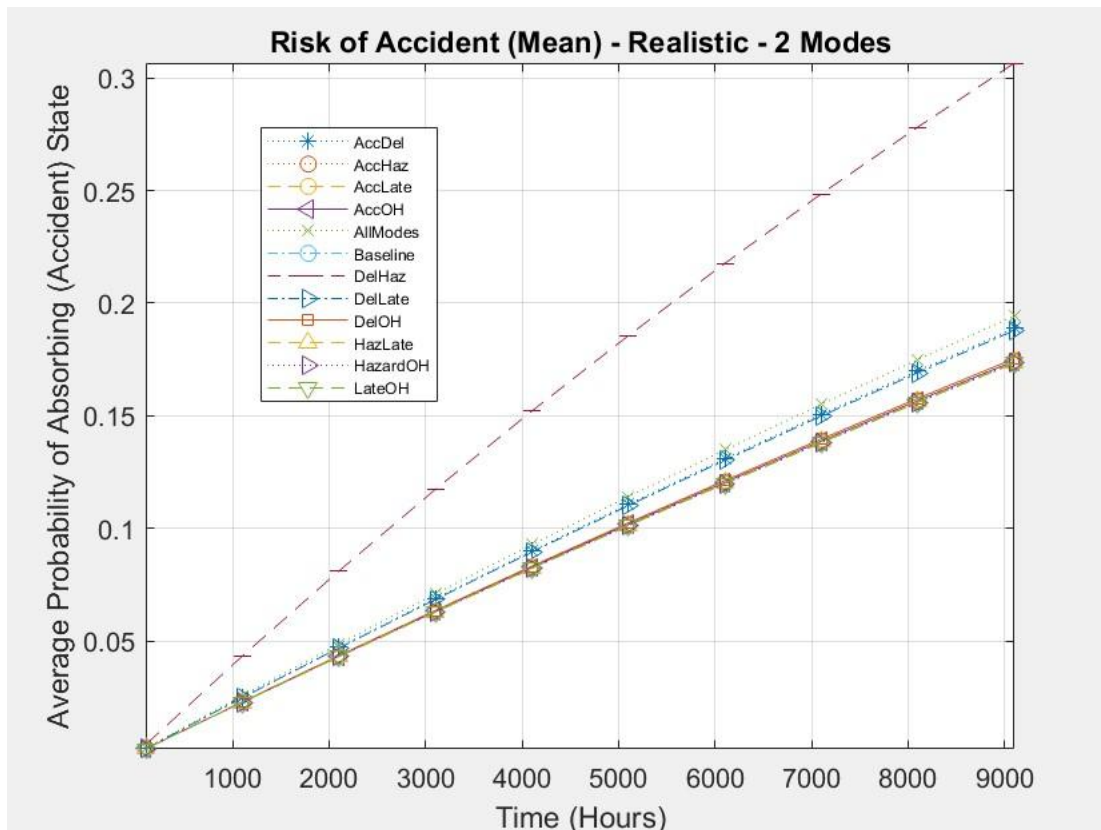
HTML generated by Möbius using the “Document” feature.

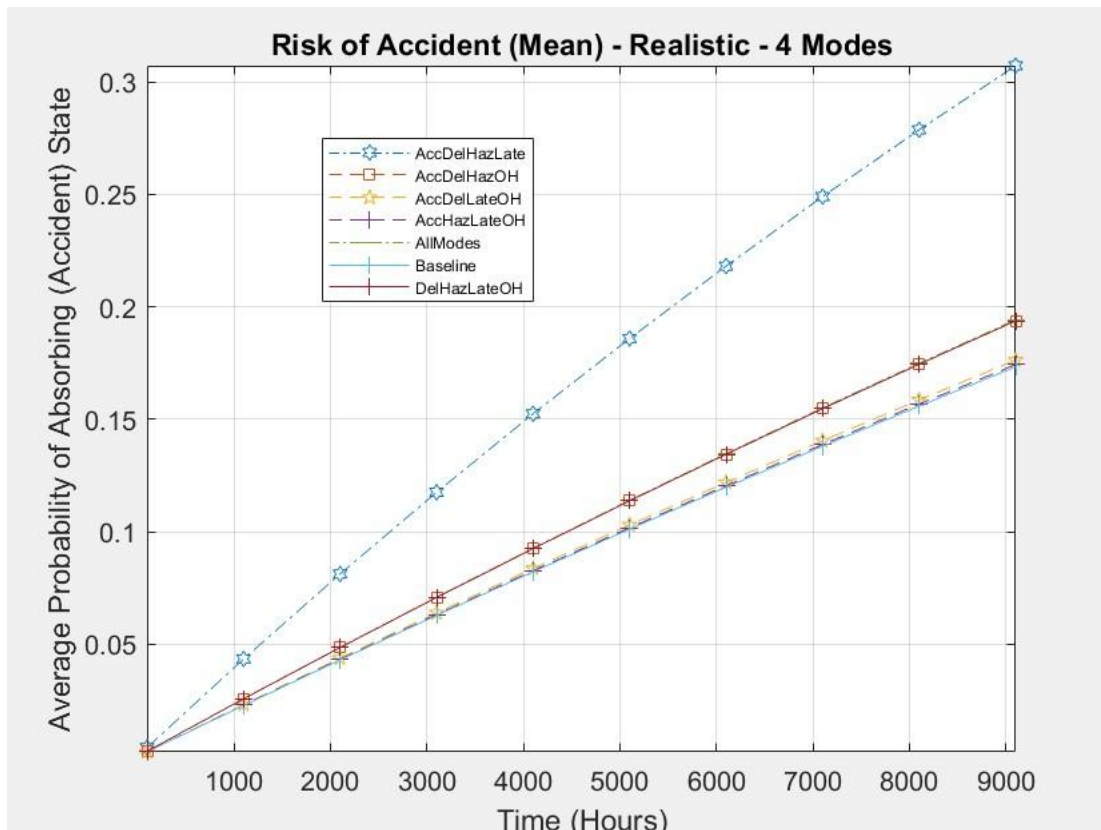
14 Appendix 3

This section holds any extra data, primarily graphs which were obtained from the model but are not strictly necessary for explanation or analysis in the main results sections.

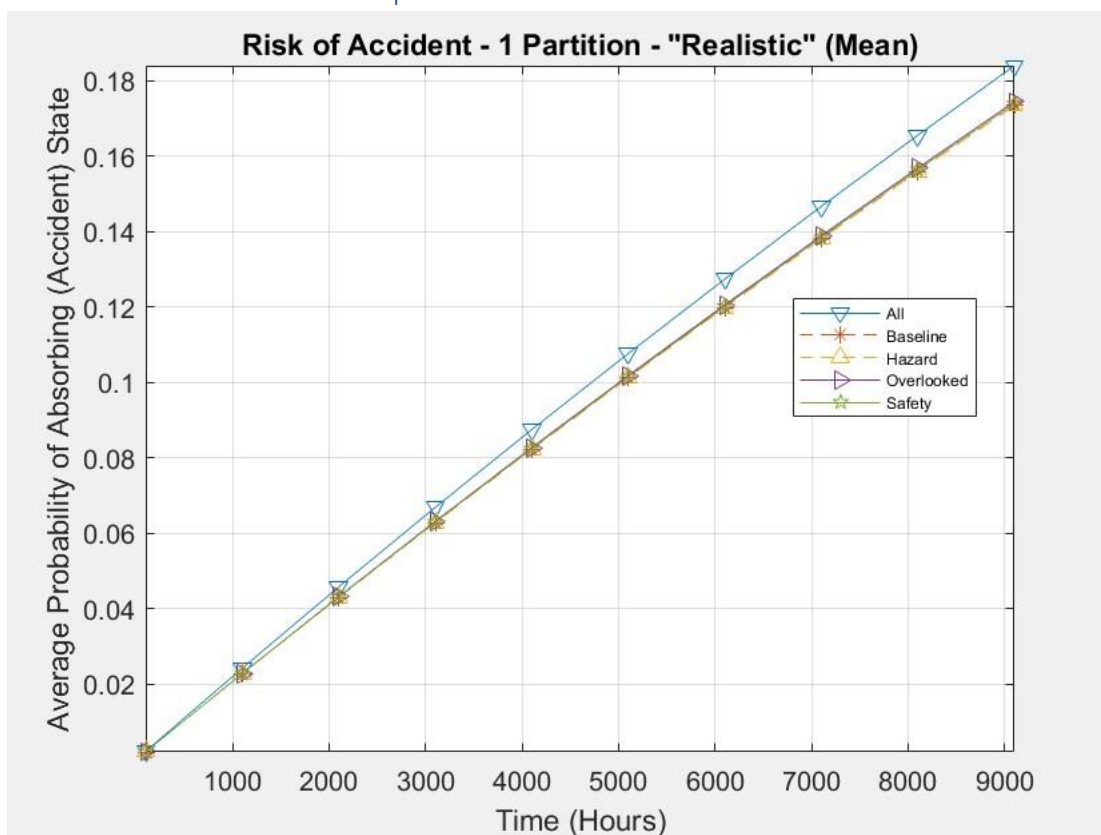
14.1 Vulnerability Mode Graphs

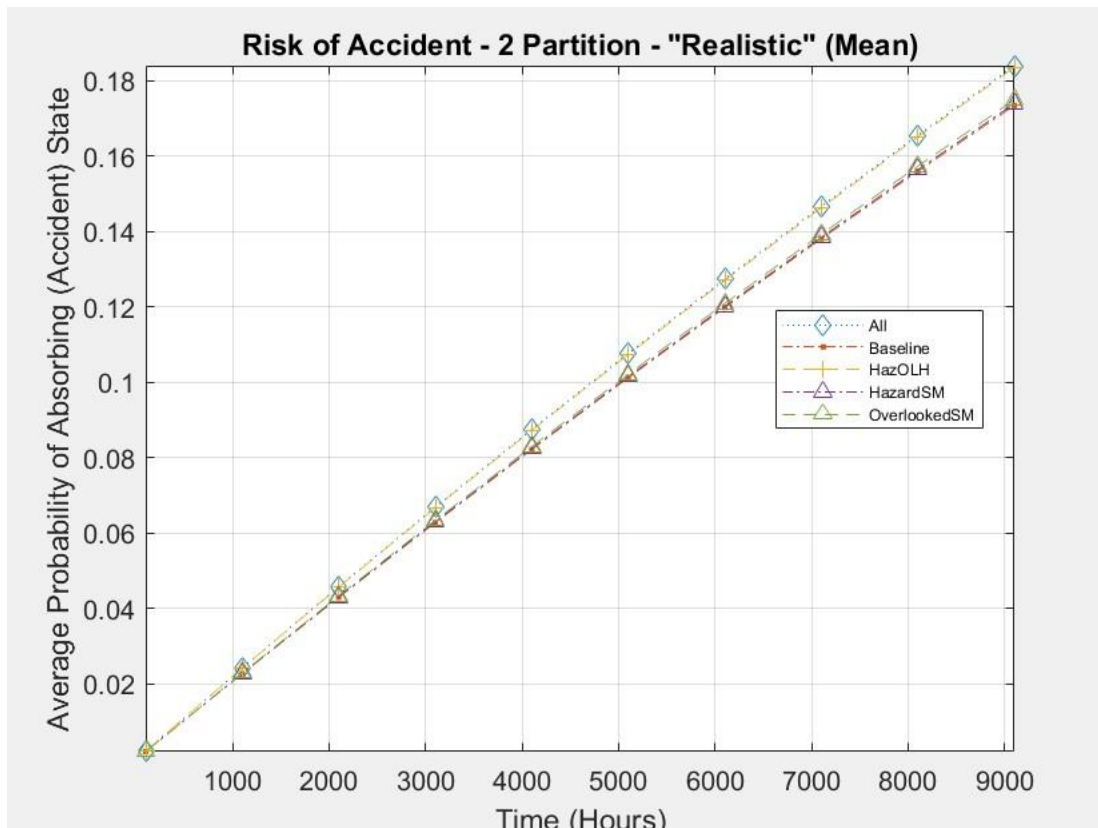






14.2 Partitioned Model Graphs





14.3 Defence-in-Depth Graphs

