



City Research Online

City St George's, University of London

Citation: Alevizos, C. (2009). SYMEX: A Systems Theory based Framework for Workflow Modelling and Execution. (Unpublished Doctoral thesis, City, University of London)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://city-ref-test.eprints-hosting.org/id/eprint/19605/>

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](#).

SYMEX: A Systems Theory based Framework for Workflow Modelling and Execution

A thesis

Submitted by

Charalampos C. Alevizos

For the Degree of

Doctor of Philosophy

At

**Centre for Human Computer Interaction Design
School of Informatics**



Supervisor: Bill Karakostas

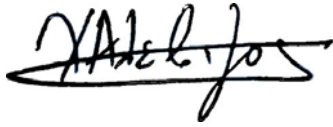
December 2009

Abstract

Workflow management systems enable organisations to deal with all aspects of business process management, including analysis, modelling, execution, and administration. Modelling workflow processes involves transformation of the process logic into a formal representation and it always remains a critical success factor for these systems. Workflow modelling languages provide constructs for capturing high-level descriptions of business processes, which are then have to be transformed and encoded into low-level execution semantics with the use of workflow programming languages. However, maintaining these models separately results in a number of issues, particularly when the various interdependencies between them are managed manually. This primarily creates difficulties in adaptation, in terms of identifying changes in high-level descriptions due to modifications of business conditions, and tracing the impact of those changes on the low-level execution semantics. Moreover, certain information included in the high-level descriptions is either partly encoded or omitted from the low-level execution semantics and at the same time, complicated business rules encoded at the execution level are not included in the high-level descriptions, creating major inconsistencies. The above issues result in high maintenance costs, reducing the overall efficiency and performance of workflow management systems. This thesis addresses the aforementioned problems by proposing a framework named SYMEX. SYMEX addresses the issue of integrating high and low-level descriptions in one unified format, from a Systems Theory perspective. SYMEX models have a mathematically defined formalism capable of capturing both high-level descriptions of business processes and low-level workflow execution semantics. Furthermore, SYMEX offers a concise and easy to learn and communicate set of constructs, allowing business analysts, process designers, and programmers to work on the same model, at different levels of abstraction. Apart from the theoretical framework, an XML-based approach for the application of SYMEX is proposed, along with a constraint-based inference engine. Additionally, SYMEX models are evaluated in terms of their complexity and prove easier to read, understand, and manage than other traditional workflow modelling approaches. However, further research is required to assess the capability of the framework, with respect to modelling workflow processes in a service-oriented environment, where activities of business processes are essentially web-services exposed on the Internet.

Declaration

The work in this thesis is based on research carried out at the Centre of Human Computer Interaction Design at City University, London, UK. No part of this thesis has been submitted elsewhere for any other degree or qualification. All work is my own unless otherwise stated.



Charalampos C. Alevizos

1st December 2009

Date

Copyright © 2009 by Charalampos C. Alevizos

"The copyright of this thesis rests with the author. No quotations should be published or information and results derived from this thesis without acknowledgement."

Acknowledgments

The major credit for the work in this thesis must go to my supervisor Dr. Bill Karakostas and to Dr. Yannis Zorgios. I am grateful for their many valuable suggestions, contributions, prompt feedback, guidance, encouragement, and support during the research. My gratitude is beyond words. I would like also to thank my examiners, Professor Neil Maiden and Dr Ilias Petrounias for their valuable comments that helped me to improve this thesis.

Many thanks to the Centre for Human Computer Interaction Design of City University, for supporting me throughout my research. I would like also to thank Bodossaki Foundation for trusting me and supporting me financially with a two-year scholarship.

Finally, I would like to thank my parents, my brother and the rest of my family, my friends and all members of Hellenic Navy's station at Corfu. Last but not least, I would like to especially thank Kyriakos, Spyros, Christos, Yannis and Stavroula. All their support and help in very different ways were crucial to the completion of this thesis.

Charalampos C. Alevizos
December 1, 2009

Contents

Contents.....	1
List of Figures	5
List of Tables	8
List of Formulas	9
1 Introduction	1
1.1 Research context.....	3
1.1.1 Process.....	3
1.1.2 Overview of Workflow technology	4
1.1.3 Workflow Process Modelling	6
1.2 Thesis motivation	8
1.2.1 Problem Statement.....	9
1.2.2 Research Hypotheses.....	9
1.2.3 Research Objectives and Questions.....	10
1.3 Research Methodology	11
1.4 Structure of the Thesis	13
1.5 Conclusions	13
2 Literature Review.....	15
2.1 Introduction	15
2.2 Model Transformation Approaches	17
2.3 Workflow Process Modelling and Execution	19
2.3.1 Workflow Process Modelling Languages.....	19
2.3.2 Workflow Process Execution Frameworks.....	38
2.4 Combining High-level representations with Low-level Execution Semantics..	48
2.4.1 UML to BPEL4WS.....	48
2.4.2 BPMN to BPEL4WS	50
2.4.3 FBPML to OWL-S	52
2.5 Enabling Disciplines	54

2.5.1	Systems Theory	55
2.5.2	Constraint-based reasoning	56
2.6	Conclusions	57
3	SYMEX: A Systems Theory based Framework for Workflow Modelling and Execution	61
3.1	Introduction	61
3.2	Requirements for an Integrated Approach	62
3.3	Theoretical Framework	64
3.3.1	The Role of Systems Theory	64
3.3.2	Definition of Concepts	65
3.3.3	Mathematical Descriptions	68
3.4	Framework Analysis	71
3.4.1	Business Semantics	71
3.4.2	Workflow Process Execution Semantics	75
3.5	Overall comparison	91
3.6	Conclusions	92
4	Application of SYMEX	93
4.1	Introduction	93
4.2	Process modelling	94
4.2.1	Process	94
4.2.2	Entity Pool	95
4.2.3	Activities	99
4.2.4	XML Schema Definition	101
4.3	Application of SYMEX modelling	103
4.3.1	Process description	103
4.3.2	SYMEX model	104
4.3.3	Process description	108
4.3.4	Discussion	113
4.4	Comparative analysis of change management	114
4.4.1	Travel agency scenario	114
4.4.2	Modelling with UML 2 Activity Diagram	115
4.4.3	SYMEX modelling	117

4.4.4	Discussion of results	119
4.5	Workflow inference engine	122
4.5.1	Theory	122
4.5.2	Constraint-based algorithm	122
4.5.3	Implementation.....	124
4.6	An Information System based on SYMEX.....	127
4.6.1	Process models	127
4.6.2	Figures	128
4.6.3	Discussion	128
4.7	Conclusions	130
5	Evaluation of the SYMEX Framework.....	133
5.1	Introduction	133
5.2	Measuring complexity of process models	134
5.3	Cross-connectivity metric (CC)	136
5.3.1	Overview	136
5.3.2	Formulas	137
5.3.3	Adapting CC metric to SYMEX process models.....	139
5.4	Calculation of metric	142
5.4.1	BPMN process model	142
5.4.2	SYMEX process model without decomposition	146
5.4.3	SYMEX process model with decomposition.....	149
5.5	Complexity Cases	155
5.5.1	Trouble Ticket Process.....	155
5.5.2	Supply Fulfilment Process.....	156
5.5.3	Hiring Process	157
5.5.4	RFQ/Order/Payment collaboration Process.....	158
5.5.5	Overall comparison	159
5.6	Conclusions	160
6	Conclusions	161
6.1	Overview	161
6.2	Summary of the Work	163

6.3 Contributions.....	165
6.3.1 Integration of high-level process descriptions and low-level workflow execution semantics into SYMEX.....	165
6.3.2 Constraint-based process modelling and execution	166
6.3.3 Application of SYMEX	166
6.3.4 Adaptation of CC metric and evaluation of SYMEX models	167
6.3.5 Publications.....	168
6.3.6 POMPEI Project	168
6.4 Research Limitations.....	169
6.5 Future Work	170
6.6 Final Remarks.....	172
References	175
Appendix – Performance of inference engine	183

List of Figures

Figure 1.1: Rain formation at a macro level	1
Figure 1.2: Venn diagram showing the research context of the thesis	3
Figure 1.3: Workflow Reference Model - Components & Interfaces [10]	5
Figure 1.4: Creating high-level descriptions and low-level executable descriptions	7
Figure 1.5: Managing high and low-level process descriptions	9
Figure 1.6: Structure of the Thesis and research methodology	12
Figure 2.1: Core elements of an IDEF0 model [86]	21
Figure 2.2: A simple IDEF0 model for a computer assembly activity [87]	22
Figure 2.3: Modelling of a business process, using EPC [94]	24
Figure 2.4: IDEF3 process description diagram [97]	26
Figure 2.5: IDEF3 Object State Transition Network Diagram [97]	27
Figure 2.6: Graphical elements of BPDs [101]	28
Figure 2.7: A travel booking process modelled with BPMN [102]	29
Figure 2.8: An XPDL sample workflow process	31
Figure 2.9: Basic notation elements of BPDM [118]	34
Figure 2.10: An example of a process model using BPDM [118]	34
Figure 2.11: An Order Request modelled with UML 2 AD [33]	37
Figure 2.12: Basic structure of BPEL4WS language [43]	40
Figure 2.13: An example of a process in BPEL4WS [136]	41
Figure 2.14: Example of a service definition in OWL-S [142]	42
Figure 2.15: An example for a Concept and Axiom Description in WSML [149]	44
Figure 2.16: Overview of mapping BPEL4WS to OWL-S [155]	47
Figure 2.17: Transformation of UML model to BPEL4WS [46]	49
Figure 2.18: A Business Process Diagram with mappings to BPEL4WS [52]	50
Figure 2.19: BPMN components and corresponding BPEL4WS translation [49, 50]	51
Figure 2.20: Mapping classes, instances and relationships between FBPML and OWL-S [53]	52
Figure 2.21: Summary of mapping FBPML and OWL-S process primitives [53]	53
Figure 2.22: Rich picture of the Workflow Technology area	57
Figure 3.1: Transforming the problem of integration using Systems Theory	65
Figure 3.2: A Systemic perspective of a workflow model	66
Figure 3.3: Visual notation of SYMEX models	66
Figure 3.4: Modelling basic aspects of business semantics	71
Figure 3.5: An example of hierarchy in a workflow process model	73
Figure 3.6: An example of feedback loop in a workflow process model	74
Figure 3.7: Workflow pattern 1: Sequence	75
Figure 3.8: Workflow pattern 2: Parallel Split	76
Figure 3.9: Workflow pattern 3: Synchronization	77

Figure 3.10: Workflow pattern 4: Exclusive Choice.....	78
Figure 3.11: Workflow pattern 5: Simple Merge.....	79
Figure 3.12: Workflow pattern 6: Multi-Choice.....	80
Figure 3.13: Workflow pattern 7: Synchronizing Merge	81
Figure 3.14: Workflow pattern 8: Multi-Merge	82
Figure 3.15: Workflow pattern 9: Discriminator	83
Figure 3.16: Workflow pattern 12: MI without Synchronization	84
Figure 3.17: Workflow pattern 13: MI with Synchronization (a).....	85
Figure 3.18: Workflow pattern 14-15: MI with Synchronization (b).....	86
Figure 3.19: Workflow pattern 16: Deferred Choice	87
Figure 4.1: Process: attributes and elements.....	94
Figure 4.2: Schema: attributes and elements	95
Figure 4.3: Control: attributes and elements	96
Figure 4.4: Mechanism: attributes and elements.....	98
Figure 4.5: Activity: attributes.....	99
Figure 4.6: Activity: elements	100
Figure 4.7: XML Schema Definition for capturing SYMEX constructs.....	102
Figure 4.8: SYMEX model - High-level description of business process.....	104
Figure 4.9: SYMEX model - Decomposition of Activity A1	105
Figure 4.10: SYMEX model - Decomposition of Activity A2	106
Figure 4.11: SYMEX model - Decomposition of Activity A3	107
Figure 4.12: Process description in XML format	112
Figure 4.13: UML 2 Activity Diagram for travel agency scenario [177]	115
Figure 4.14: Adapted UML 2 Activity Diagram for travel agency scenario [177]	116
Figure 4.15: SYMEX model for travel agency scenario	117
Figure 4.16: Adapted SYMEX model for travel agency scenario	118
Figure 4.17: Changes to UML 2 Activity Diagram for the travel agency scenario	120
Figure 4.18: Changes to SYMEX model for the travel agency scenario	121
Figure 4.19: UML Class Diagram of the implemented workflow inference engine	124
Figure 4.20: UML Class Diagram: Activities and Activity classes.....	125
Figure 4.21: UML Class Diagram: Schemas and Schema classes.....	125
Figure 4.22: UML Class Diagram: Mechanisms and Mechanism classes	125
Figure 4.23: UML Class Diagram: Controls and Control classes.....	126
Figure 4.24: Database diagram of the implemented workflow inference engine.....	126
Figure 5.1: Reading a SYMEX model and calculating the CC metric.....	141
Figure 5.2: BPMN process model [49].....	142
Figure 5.3: SYMEX model without decomposition.....	146
Figure 5.4: Level 1 of SYMEX model	149

Figure 5.5: Level 2 of SYMEX model: decomposed Activity “questionnaire”	149
Figure 5.6: Level 2 of SYMEX model: decomposed Activity “complaint”	150
Figure 5.7: Level 1 of SYMEX model with calculated CC metric for the composite Activities	152
Figure 5.8: BPMN model: Trouble Ticket Process [199]	155
Figure 5.9: BPMN model: Supply Fulfilment Process [200]	156
Figure 5.10: BPMN model: Hiring Process [201]	157
Figure 5.11: BPMN model: RFQ/Order/Payment collaboration Process [202]	159
Figure 0.1: Phase 1’s execution time (ms)	185
Figure 0.2: Phase 2’s execution time (ms) for a typical Activity	185
Figure 0.3: Phase 2’s execution time (ms) depending on execution Mechanisms	186
Figure 0.4: Phase 3’s execution time (ms)	186
Figure 0.5: Phase 4’s execution time (ms)	187
Figure 0.6: Phase 4 – Execution time (ms) based on the number of Inputs	187
Figure 0.7: Phase 4 – Execution time (ms) of checking an Activity’s state	187
Figure 0.8: Total Execution time (ms) using a typical scenario	188
Figure 0.9: Total Execution time (ms) using an extreme scenario	188

List of Tables

Table 2.1: Categorization of Process Execution Frameworks	38
Table 2.2: UML Profile and BPEL mapping [48].....	48
Table 2.3: Main limitations identified in the literature	59
Table 3.1: Workflow patterns [104]	63
Table 3.2: Comparison based on workflow patterns [176]	89
Table 3.3: Overall comparison of SYMEX to other approaches.....	91
Table 4.1: Pseudo-code listing of constraint-based algorithm.....	123
Table 4.2: List of modelled Processes	127
Table 4.3: Figures from the Information System based on SYMEX.....	128
Table 5.1: Complexity metrics for software and business process models [196].....	135
Table 5.2: Calculation of weight of nodes for BPMN process model	143
Table 5.3: Calculation of weight of arcs for BPMN process model	144
Table 5.4: Calculation of weight of connections for BPMN process model.....	145
Table 5.5: Calculation of weight of nodes for SYMEX process model.....	147
Table 5.6: Calculation of weight of arcs for SYMEX process model	147
Table 5.7: Calculation of weight of connections for SYMEX process model	148
Table 5.8: Calculation of weight of nodes for Activity N2,3,4,5 of SYMEX model	150
Table 5.9: Calculation of weight of arcs for Activity N2,3,4,5 of SYMEX model.....	150
Table 5.10: Calculation of weight of connections for Activity N2,3,4,5 of SYMEX model.....	151
Table 5.11: Calculation of weight of nodes for Activity N6,7,8 of SYMEX model	151
Table 5.12: Calculation of weight of arcs for Activity N6,7,8 of SYMEX model.....	151
Table 5.13: Calculation of weight of connections for Activity N6,7,8 of SYMEX model.....	152
Table 5.14: Calculation of weight of nodes for Level 1 of SYMEX model.....	153
Table 5.15: Calculation of weight of arcs for Level 1 of SYMEX model	153
Table 5.16: Calculation of weight of connections for Level 1 of SYMEX model	153
Table 5.17: Calculation of CC-metric for all the cases.....	159

List of Formulas

Formula 3.1: Definition of $I(A)$ – Inputs of Activity	68
Formula 3.2: Definition of $O(A)$ – Outputs of Activity	68
Formula 3.3: Definition of $C(A)$ – Controls of Activity	68
Formula 3.4: Definition of $M(A)$ – execution Mechanisms of Activity	68
Formula 3.5: Rule 1	69
Formula 3.6: Rule 2	69
Formula 3.7: Rule 3	69
Formula 3.8: Rule 4	69
Formula 3.9: Rule 5	69
Formula 3.10: Rule 6	69
Formula 3.11: Definition of Activity States for a workflow process Σ	70
Formula 3.12: Definition of status 'available' for Activity A_i	70
Formula 3.13: Definition of execution Mechanism of Activity A_i	70
Formula 3.14: Definition of Control of Activity A_i	70
Formula 3.15: Hierarchy's semantics	72
Formula 3.16: Feedback loop's semantics	74
Formula 3.17: Semantics of Workflow pattern 1: Sequence	75
Formula 3.18: Semantics of Workflow pattern 2: Parallel Split	76
Formula 3.19: Semantics of Workflow pattern 3: Synchronization	77
Formula 3.20: Semantics of Workflow pattern 4: Exclusive Choice	78
Formula 3.21: Semantics of Workflow pattern 5: Simple Merge	79
Formula 3.22: Semantics of Workflow pattern 6: Multi-Choice	80
Formula 3.23: Semantics of Workflow pattern 7: Synchronizing Merge	81
Formula 3.24: Semantics of Workflow pattern 8: Multi-Merge	82
Formula 3.25: Semantics of Workflow pattern 9: Discriminator	83
Formula 3.26: Semantics of Workflow pattern 14: MI with Synchronization (a)	86
Formula 3.27: Semantics of Workflow pattern 14-15: MI with Synchronization (b)	86
Formula 3.28: Semantics of Workflow pattern 16: Multi-Choice	87
Formula 4.1: Control of Activity A_3 at the initial SYMEX model	117
Formula 4.2: Control of Activity A_3 at the adapted SYMEX model	119
Formula 4.3: Definition of Activity States for a business process Σ	122
Formula 4.4: Definition of status 'available' for Activity A_i	122
Formula 4.5: Definition of execution Mechanism of Activity A_i	122
Formula 4.6: Definition of Control of Activity A_i	122

**THE FOLLOWING PARTS OF THIS THESIS HAVE BEEN REDACTED
FOR COPYRIGHT REASONS:**

Figure 2.11: An Order Request modelled with UML 2 AD [33]	37
Figure 5.8: BPMN model: Trouble Ticket Process [199]	155
Figure 5.10: BPMN model: Hiring Process [201]	157

1 Introduction

Physical phenomena are the outcome of processes that consist of a number of tasks executed in some pre-specified or random order. The difficulty of capturing and defining the tasks depends on the conceptualisation process required to describe the phenomena. If we would try to describe the rain phenomenon at a macro level, we would say that it is a process consisting of a number of tasks, in which the sun heats the water in oceans, rivers and lakes, water is vaporised due to heat, condenses into drops when vapours reach a certain level on concentration and form rain. Eventually water returns to oceans, rivers, and lakes. The process of rain phenomenon described, always follows the same cyclic order of tasks, as shown in Figure 1.1.

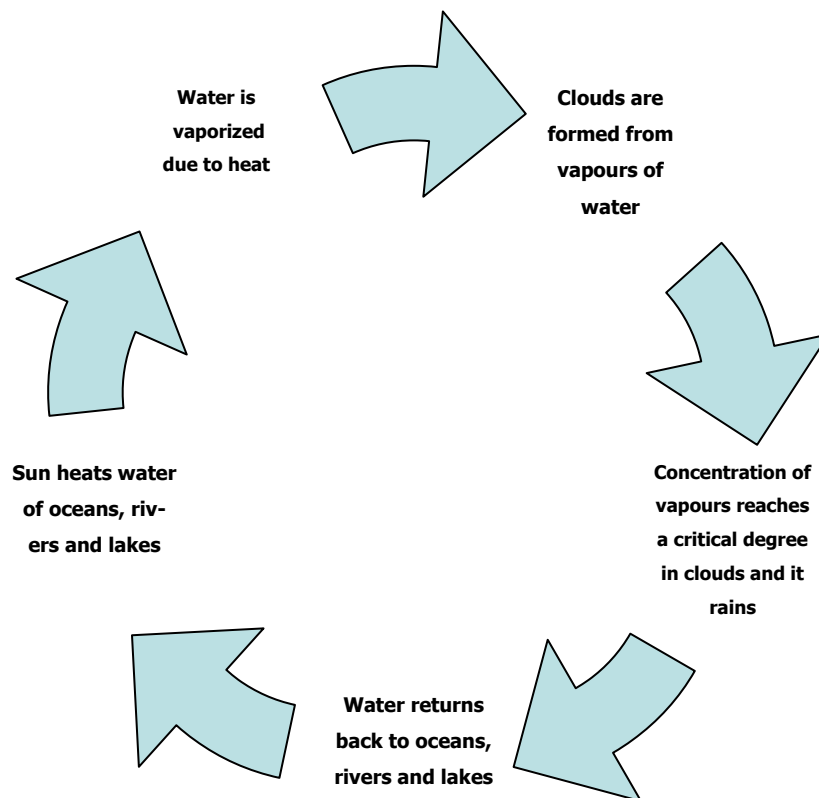


Figure 1.1: Rain formation at a macro level

Similar to physical phenomena, human activities are also processes consisting of a number of tasks realized in some order. Among human activities, business activities are diachronic; trade was the first form of business activity originated with the start of communication in prehistoric times, and since then, business activities cover all aspects of human life. However, unlike physical phenomena, which generally follow a specific pattern of developmental stages, business activities often change adapting to new business needs in a more dynamic manner. This increases the level of complexity, making the process of rationalising and managing the changes manually a difficult task. In order to facilitate the process of modelling and execution of business processes, specific technological solutions have been developed, known as workflow management systems. In gen-

eral, workflow technology is a research area dealing with all aspects of business process management, including analysis, modelling, execution, and administration. All these aspects are also referred in the literature as workflow management.

As in most engineering processes, the success of a workflow management system relies much on the modelling and the design phase of business processes [1, 2], [3, 4]. There is extensive research attempting to formulate solutions that help business analysts to deliver workflow applications that adapt to changing business conditions. The research mainly focuses on providing modelling tools and languages for conceptualising and thus modelling the tasks of a business process at two separate levels; high structural level and low execution level. However, during the adaptation of a business process, business analysts have to maintain manually the consistency between the two levels of process description, which in turn creates difficulties in designing adaptable workflow management solutions.

This thesis mainly focuses on the modelling aspect of business processes within a workflow management context (i.e. workflow process modelling). Among the open issues of modelling, the problem of maintaining two different models for high-level representations and low-level execution semantics is identified as the main problem of our research. The research objectives include the identification of the relationships between the two descriptions and their integration into a united form.

The structure of this Chapter is as follows. Section 1.1 defines the research context of the thesis, introducing the concept of process, giving an overview of workflow technology area and focusing on the open issues of workflow process modelling. Section 1.2 discusses the thesis motivation, stating the research problem, and posing the research questions and objectives. Section 1.3 describes the research methodology and Section 1.4 presents the structure of the thesis. Finally, Section 1.5 concludes the Chapter.

1.1 Research context

Venn diagram in Figure 1.2 shows the research context of the thesis, which focuses on the modelling aspects of business processes for the purpose of workflow management. More specifically, we are interested in investigating the open issue of maintaining consistency between separate high and low-level workflow process models.

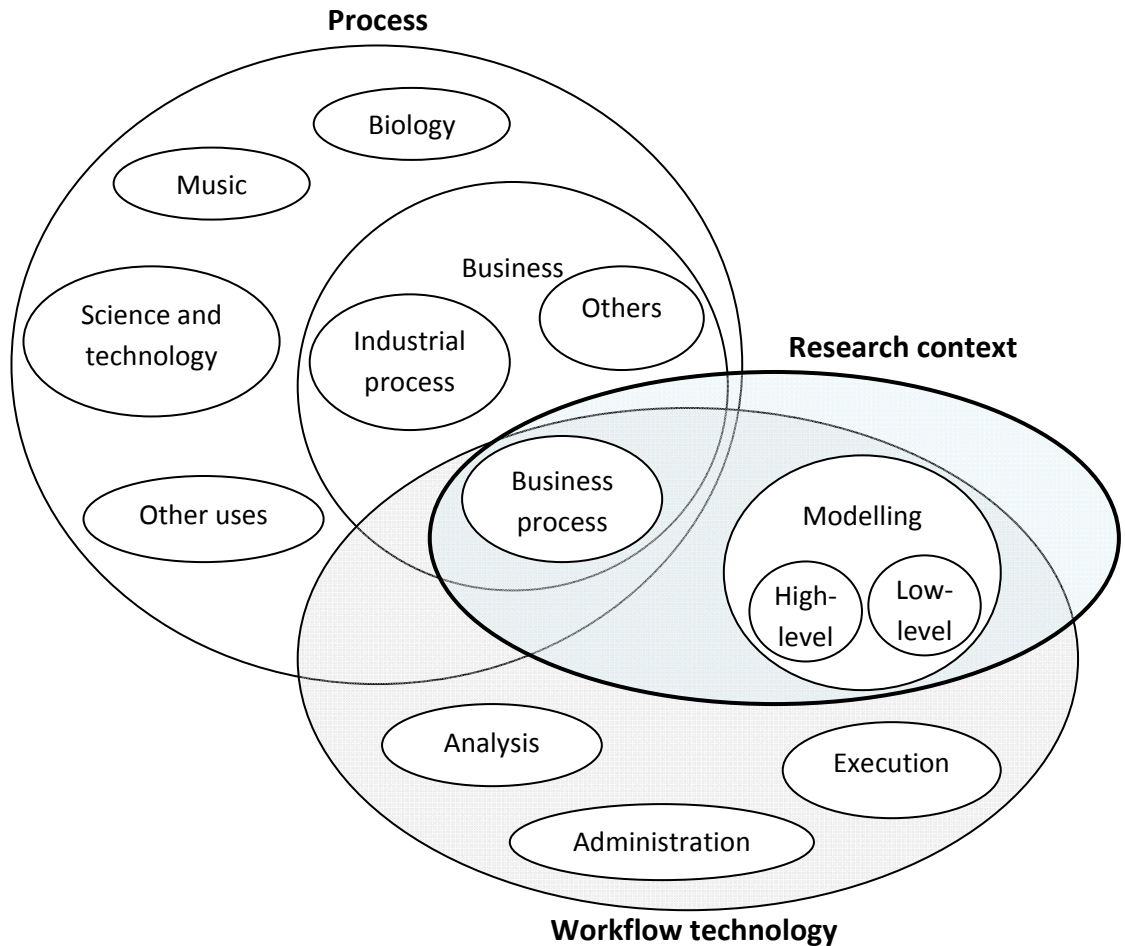


Figure 1.2: Venn diagram showing the research context of the thesis

1.1.1 Process

Process is a general term that applies to many aspects of life. Therefore, the term may refer to a number of disciplines, e.g. a biological process, a chemical process, an industrial process, a business process etc. In this Thesis, we refer to business process. In 1993, Davenport gave a definition of a business process as *"a structured, measured set of activities designed to produce a specific output for a particular customer or market. It implies a strong emphasis on how work is done within an organization, in contrast to a product focus's emphasis on what. A process is thus a specific ordering of work activities across time and space, with a beginning and an end, and clearly defined inputs and outputs: a structure for action"* [5].

A business process can have several levels of decomposition. Decomposition captures separation of concerns at the business level, revealing the relations between the different parts of a process and providing a natural way for business people to categorize and make hierarchy of their business processes. During process design, a business process can be decomposed into several sub-processes. Each sub-process has its own characteristics and can be further decomposed, but eventually contributes to achieving the goal of the parent process [6]. The levels of decomposition depend on the size of the process and the levels of details that the process designer needs to expose in the model. Finally, the success of process model depends not only on the process designer but also on the modelling technique that is used. Choosing a technique is an important decision, as there are many proposals for business process modelling within the workflow context, with different characteristics.

1.1.2 Overview of Workflow technology

Workflow technology has appeared in a primitive form in the early 1970's, supporting image and document-routing automation in enterprises. One of the first attempts to define the term "workflow", was made by the Workflow Management Coalition [7] in 1996, according to which workflow is *"the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules"* [8]. Workflow technology has evolved a lot since then, and during the last 10 years, found its way in the support of model-driven development and the management of business process realization, in application areas such as [9]:

- packaged applications, as a means of customization,
- electronic commerce, to coordinate an enterprise's interaction with its customers over the Internet,
- business objects for composing workflow-process components, message-broker environments, processing of complex requests, and
- virtual enterprises, to coordinate inter-enterprise processes.

As many vendors and developers began to build their own workflow applications in order to implement workflow processes, there was a need for workflow standards. Workflow Management Coalition identified those parts of workflow applications and workflow-management systems that should be standardized [9] and introduced Workflow Reference Model [10]. Figure 1.3 illustrates the major components and interfaces within the workflow architecture as defined in the Workflow Reference Model.

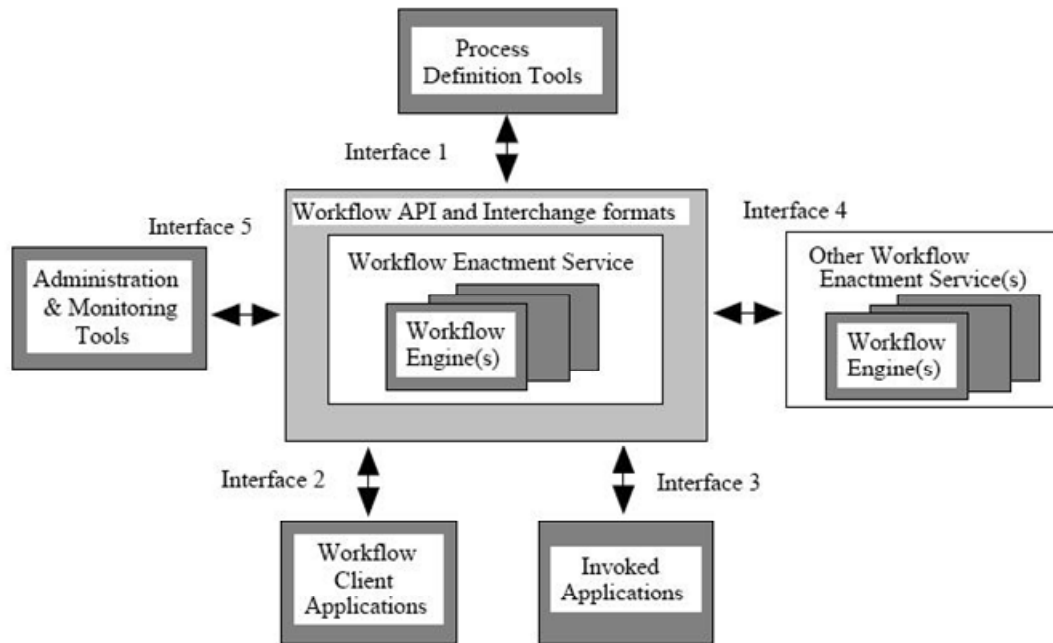


Figure 1.3: Workflow Reference Model - Components & Interfaces [10]

While business environments have become exceedingly dynamic and competitive since the late 90's, workflow systems did not provide the flexibility necessary to support the dynamic nature of business processes [11]. Research about issues and challenges related to managing change and time in workflows representing dynamic business processes [11], provided a number of feasible approaches to tackle dynamically changing workflow processes. Aalst in [12] suggested the use of generic process models; the generic process model extended the classical workflow models, primarily based on routing diagrams, with inheritance diagrams. This allowed for the specification of process families composed of variants and addressed two main problems related to adaptive workflow: (1) providing management information at the right aggregation level, and (2) supporting dynamic change, i.e., migrating cases from an old to a new workflow.

The next step in the workflow area was the need to support cross-enterprise workflows. This need was addressed by researchers, such as Zeng et al. in [13] and Wang in [14], with the use of agent-based approaches. Other researchers employed Common Object Request Broker Architecture (CORBA) [15] to enable distributed workflow systems. Purvis et al. in [16], propose a framework that employs Petri nets to model the interaction between various sub-processes. Other approaches, such as the RainMan system [17], started using open standards and Web-browser based user interface components. The importance of open standards in decentralized workflow execution and disconnected participation, made the research community consider the use of Internet-based standards such as HTTP protocol and XML, as Hayes et al. discuss in [18]. Web-based workflow management systems soon made their appearance. This kind of systems support a number of web protocols, such as HTTP, XML and XSL and they use HTTP as the core communication protocol. Characteristic examples are Magi [19] and Workspaces [20].

Lately, following the recent requirements and needs, workflow technology is adopting the use of web-services [21]. Web-services serve as an infrastructure providing seamless interoperability among networked workflow processes. Thus, web-services enable workflow technology to move a step towards interoperability, thanks to their ability of reducing heterogeneity through standardized interaction paradigms [22]. The presence of web-services raised issues of monitoring and controlling their execution, to ensure they operate with the desired quality levels. The description and invocation of web-services have been addressed by Web-services Description Language (WSDL) [23] and Simple Object Access Protocol (SOAP) [24] respectively. In addition, standardization efforts such as WS-Transaction, WS-Specification, WS-Coordination, WS-Addressing [25-28] are trying to address the transaction, coordination, addressing and discovery issues. A number of models have also been proposed in order to support the management of complex interactions between clients and web-services, like the model of Ardissono et al. in [29].

In summary, workflow technology evolved from simple document routing automation to distributed workflow management systems, supporting the management of business process realization across enterprises. Modelling of processes is considered as one of the most crucial factors for the success of a workflow management system and it is discussed in the next Section.

1.1.3 Workflow Process Modelling

The success of a workflow management system and thus its capability to adapt to changing business conditions relies heavily on the success of the process modelling phase. The common practice, as far as the business process modelling is concerned, includes the following steps:

1. Analysts analyze and capture the requirements of the information system that will be developed. This step is equivalent with the requirements analysis in software engineering, which is one of the most important challenges in order to rationalize the processes from requirements definition to design [30]. Requirements analysis helps to determine the needs or conditions to meet for a workflow process, taking account of the possibly conflicting requirements of the various stakeholders, such as beneficiaries or business users [31]. The captured requirements must be actionable, measurable, testable, related to identified business needs, and defined to a level of detail sufficient for the workflow process design [32].
2. Designers, based on the analysis, design a high-level representation of the business process using modelling techniques, such as UML-based diagrams [33].
3. Then programmers, in collaboration with designers, encode the business process in low-level workflow execution semantics, using execution frameworks such as BPEL4WS [34].

Figure 1.4 shows an example of modelling a business process of a hospital. Specifically, analysts first create a high-level representation of a business process that takes place inside the hospital.

Analysts capture the requirements of the process, i.e. interacting roles, tasks of the process, information flow across the process etc. Then, designers create a UML Activity Diagram [33], translating the high-level representation using constructs and patterns of UML. Finally, programmers use the programming constructs offered by BPEL4WS, in order to transform the UML Activity Diagram into code that will be used by a workflow execution engine to automate the business process.

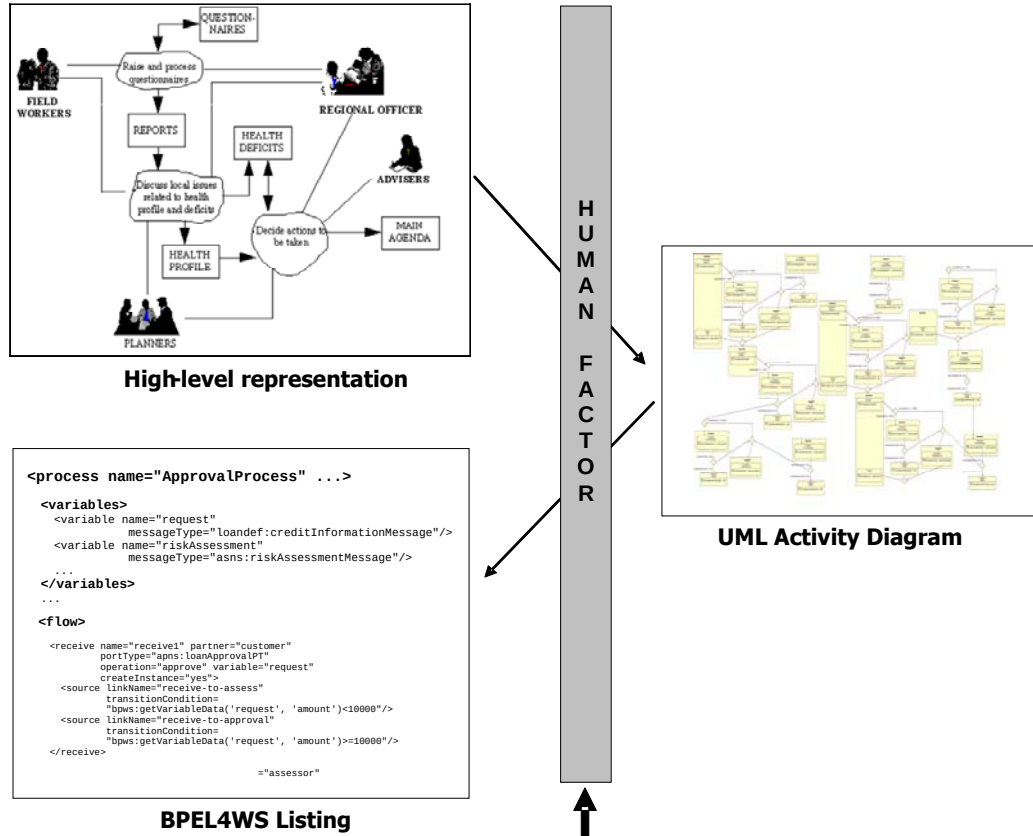


Figure 1.4: Creating high-level descriptions and low-level executable descriptions

It is common practice to maintain manually the separate models of high-level representations and low-level execution descriptions. This situation usually leads business analysts and process designers to hard code certain semantics and elements of the business process environment to either the high-level or the low-level description. In case of adaptation, changes have to be first identified in the high-level representations and then needs to be adapted in the low-level descriptions. These transformations rely much on the human factor and often a number of inconsistencies between the two models appear. Such inconsistencies increase the level of complexity of the workflow process modelling activity, which becomes even more difficult to adapt under the continually changing business conditions and may lead into a malfunctioning workflow management system.

Workflow process modelling therefore is one of the most critical phases for the success of a workflow management system. Up to now, there is not much research towards an integrated approach that would combine high-level representations of business processes with low-level executable descriptions. For that reason, the need of an integrated framework, which would be capable of effectively encoding processes in high-level descriptions as well as their execution semantics in low-level workflow descriptions, is evident.

1.2 Thesis motivation

As workflow technology evolves, a landscape of languages and techniques for workflow process modelling has emerged and it is continuously being enriched with new proposals from different vendors and coalitions. Inside this landscape of languages and techniques, there is not much research towards an integrated approach that would combine high-level representations of business processes with low-level executable descriptions.

Business process models are representations defined in high-level descriptions by designers, while executable business process descriptions are computer interpretable descriptions automating the execution of business processes, defined in low-level descriptions by developers. The two are maintained as completely different models. Therefore, when there is a business change demand, designers have to understand and realise the changes in the high-level description of the business process. Then developers need to propagate the changes again on the low-level execution semantics of the process. This transformation is commonly done manually, relying on the capability of analysts and designers to trace the changes from the high-level model to the low-level description and vice versa. Moreover, different type of information between the two models often results to inconsistency. Specifically, designers often ignore hard coded business rules at execution semantics and on the other hand, developers overlook certain information modelled at a high-level of abstraction.

1.2.1 Problem Statement

High-level process descriptions and low-level workflow execution semantics are maintained as two completely different models and the relationships between them are managed manually (Figure 1.5). This primarily creates difficulties in adaptation, in terms of identifying changes in high-level descriptions due to modifications of business conditions, and tracing the impact of those changes on the low-level execution semantics. Moreover, certain information modelled in the high-level descriptions is either partly encoded or omitted from the low-level execution semantics and at the same time, complicated business rules encoded at the execution level are not included in the high-level descriptions, creating major inconsistencies. The above issues result in high maintenance costs reducing the overall efficiency and performance of workflow management systems.

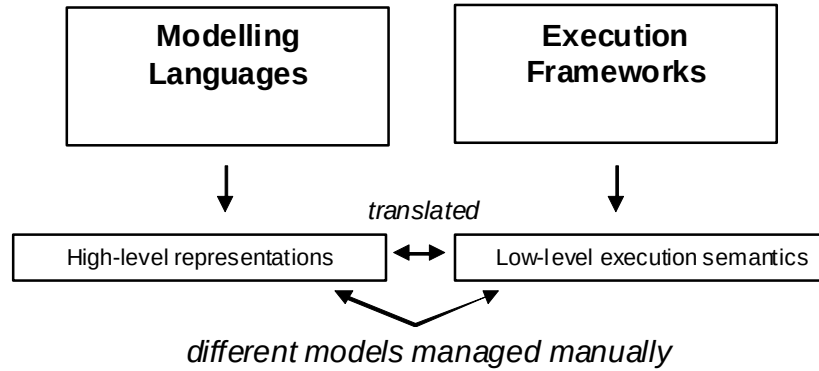


Figure 1.5: Managing high and low-level process descriptions

1.2.2 Research Hypotheses

We argue that an integrated workflow modelling and execution framework for capturing and encoding process semantics can be defined directly from first principles of Systems Theory. Specifically, within the scope of our research, we argue that:

1. Systems oriented constructs such as scope, activity, input, output, transformation mechanisms, control logic, and feedback loops, together with the concept of decomposition provide the necessary means to capture and represent systemic views of complex business processes.
2. A systems oriented workflow modelling and execution language can provide an integrated framework, at the right level of abstraction, upon which process descriptions can be defined in a computer-interpretable way, in order to support the execution of business processes.
3. Systems oriented process semantics are easier to understand and manage than existing ones based on BPMN, UML activity diagrams etc.

1.2.3 Research Objectives and Questions

The research objectives (RO) and the research questions (RQ) of the thesis are as follows:

RO1 Integration of high-level process descriptions and low-level workflow execution semantics

RQ1.1 How can we enable integration of high-level process descriptions and low-level workflow execution semantics in the SYMEX approach using the main principles of Systems Theory?

RQ1.2 What kind of conceptual constructs of Systems Theory can be used to model high-level and low-level workflow process descriptions?

RQ1.3 How can we formally describe systems oriented process models that encode both high-level and low-level workflow process descriptions?

RO2 Application of the integrated workflow process modelling approach

RQ2.1 How can systems oriented process descriptions be defined in order to model real life business workflows?

RQ2.2 What is required in order to execute systems oriented business workflows?

RO3 Assess the effectiveness in SYMEX models compared to existing available techniques

RQ3 What kind of measures can we define in order to assess effectiveness; how easy is to understand and manage changes in SYMEX models, compared to existing available techniques?

1.3 Research Methodology

The research methodology followed in the thesis comprises the following eight steps:

1. Research scope

Definition of the research context of the thesis.

2. Problem statement and Research Objectives

Definition of the thesis motivation, which includes the problem statement and the research objectives and questions.

3. Literature Review

Review and analysis of the literature in workflow process modelling and execution, identifying strengths and weaknesses. Also, review of disciplines that enable the solution of the research problem.

4. Introduction of the proposed framework

Definition of the basic concepts and constructs of the proposed framework.

5. Formalization of the proposed framework

Definition of the formal semantics of the proposed framework.

6. Application of framework

Proposal of an implementation approach in order to design and develop a prototype of the framework.

7. Evaluation of framework

Evaluation of the framework in order to assess its capabilities, strengths, weaknesses, and possible improvements.

8. Overall conclusions

Discussion of the contributions of the research, possible limitations, and future research directions.

Figure 1.6 presents the research methodology as flow chart, along with the structure of the thesis.

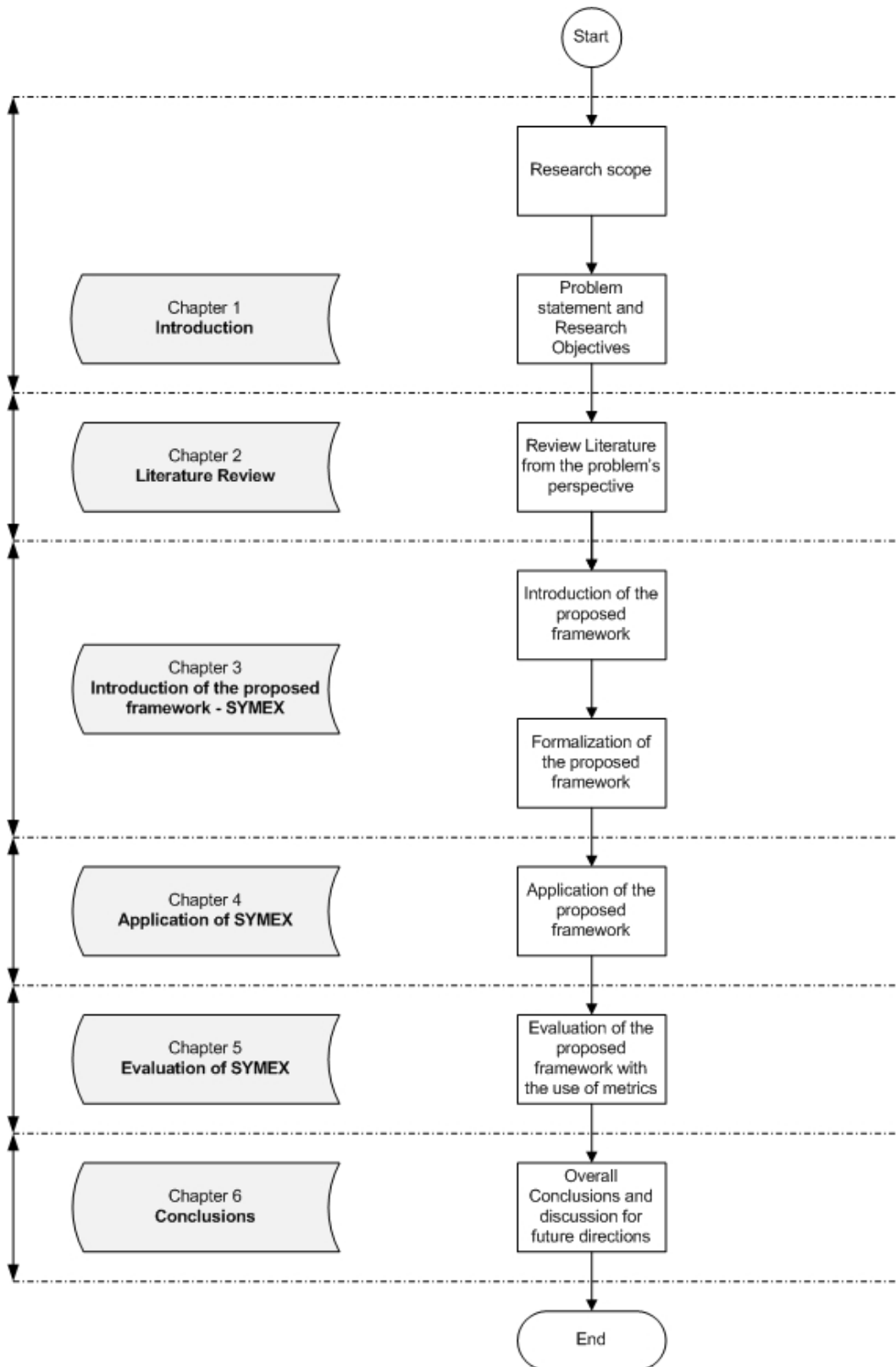


Figure 1.6: Structure of the Thesis and research methodology

1.4 Structure of the Thesis

The structure of the thesis is as follows:

Chapter 1 defines the research context of the thesis, along with problem statement, the research objectives, and questions.

Chapter 2 reviews the literature of workflow process modelling and execution frameworks from the problem's perspective. In this Chapter, we identify strengths and weaknesses of current approaches and review disciplines that enable the solution of the research problem.

Chapter 3 addresses research objective **RO1** by integrating high-level process descriptions and low-level workflow execution semantics into a unified format, using a Systems Theory based approach. In this Chapter, we also define the basic concepts of the proposed framework (SYMEX), its constructs, and formal semantics.

Chapter 4 addresses research objective **RO2**. Specifically, it proposes an implementation approach in order to design and develop a prototype of the framework. This Chapter also demonstrates how the proposed approach facilitates and accelerates adaptation of processes, in comparison with approaches that explicitly sequence activities of processes. Additionally, it presents a proposed implementation of a workflow inference engine that can execute SYMEX models.

Chapter 5 evaluates effectiveness of SYMEX by employing complexity metrics for business process models. These metrics quantify the benefits of the framework and address research objective **RO3**.

Chapter 6 concludes our research. The major contributions, along with the limitations and future work directions are discussed.

1.5 Conclusions

In this Chapter we defined the research context of the thesis; business processes and modelling within the workflow context. We introduced the concept of business process, gave an overview of the workflow technology area, and focused on the importance of the workflow process modelling. Then we defined the problem statement of the thesis together with the research objectives and questions. Finally, we described the research methodology and the structure of the thesis.

2 Literature Review

2.1 Introduction

In Chapter 1, we concluded that relationships between high-level process descriptions and low-level workflow execution semantics are maintained manually, resulting in high maintenance costs and processes that cannot adapt easily and quickly to changes, consequently reducing the overall efficiency and performance of workflow management systems. In this Chapter, we discuss the model transformation approaches that have been reported in the software engineering literature and how they can affect our research and we then review the workflow process modelling languages and execution frameworks that are mainly used by modellers and process designers to capture the high-level descriptions and the low-level execution semantics respectively.

There are a number of different approaches for workflow process modelling, including graphs of events and functions [35], function modelling [36], process flow and object state transition networks [37], flowcharting techniques [38], XML-based design formats [39] and activity diagrams [40]. All workflow process modelling languages focus on the creation of high-level descriptions using a set of visual constructs [35], [36], [37], [38], [40]. On the other hand, they lack low-level constructs and precise semantics for the realization of business processes, in terms of extracting low-level process execution details. Thus, process designers use process execution frameworks in order to translate the abstract layer of modelling to the execution layer of processes. Workflow process execution frameworks are used to define low-level execution semantics for the processes that will be eventually executed by an execution engine. There are two main categories of execution frameworks; mathematical-based [41], [42], [43] and ontology-based [44], [45]. BPEL4WS and OWL-S are considered as the major representatives of each category respectively. Due to the distance between modelling and execution, there are some research approaches proposing mappings, in order to reduce the gap between high-level process descriptions and low-level workflow execution semantics. Among them, there are approaches using mathematical-based [46-48], [49-52] and those using ontology-based [53] modelling and execution frameworks. Finally, this Chapter discusses the enabling disciplines of our research: Systems Theory and Constrained-based reasoning. These research areas enable us to address the research objectives we have posed in Chapter 1.

The structure of this Chapter is as follows. Section 2.2 discusses model transformation approaches and examines in what extent they can be adopted in our research. Section 2.3 outlines and reviews workflow modelling languages and process execution frameworks. Section 2.4 reviews research approaches combining high-level modelling and execution semantics. Section 2.5 introduces the enabling disciplines for our proposed framework. Finally, Section 2.6 concludes the

Chapter with an overview of the research area of workflow process modelling and execution and a discussion of the identified limitations.

2.2 Model Transformation Approaches

One of our main research objectives is to maintain consistency between workflow models at different levels of abstraction. This Section discusses the model transformation approaches that have been reported in the software engineering literature and examines in what extent they can be adopted in our research.

Model-Driven Architecture (MDA) [54] has been proposed by Object Management Group (OMG) [55] as an approach to software development based on modeling and automated mapping of models to implementations. The basic idea involves the definition of a platform independent model (PIM) and its automated mapping to one or more platform-specific models (PSMs). The benefits of such an approach are similar with our research objectives within the workflow context and include [56]:

1. improved portability due to separating the application knowledge from the mapping to a specific implementation technology,
2. increased productivity due to automating the mapping,
3. improved quality due to reuse of well proven patterns and best practices in the mapping,
4. improved maintainability due to better separation of concerns and
5. better consistency and traceability between models and code.

In 2002 there was an effort to define a foundation for transforming PIMs into PSMs and OMG initiated a standardization process by issuing a Request for Proposal (RFP) on Query / Views / Transformations (QVT) [57]. Driven by the OMG's request and by practical needs, a large number of approaches to model transformation have been proposed. A classification of existing model transformation approaches based on [56] is the following:

- **direct manipulation approaches;** offer an internal model representation plus some API to manipulate it and are usually implemented as an object-oriented framework, e.g. Jamda [58], JMI [59]
- **relational approaches;** this category groups declarative approaches where the main concept is to state the source and target element type of a relation using constraints in terms of mathematical relations, e.g. [60], [61], [62], [63], [64]
- **graph-transformation-based approaches;** based on the theoretical work on graph transformations, e.g. [65], VIATRA [66], ATOM [67], GreAT [68], UMLX [69], and BOTL [70].
- **structure-driven approaches;** have two distinct phases: the first phase is concerned with creating a hierarchical structure of the target model and the second phase sets the attributes and references in the target, e.g. [71]
- **hybrid approaches;** combine different techniques from the previous categories, e.g. Transformation Rule Language (TRL) [72], Atlas Transformation Language (ATL) [73]

It is a fact that model transformation approaches shows a direction in how to transform one model into another but they cannot be used as such in order to address our research objectives. In this thesis, we deal with business process-based information and characteristics, which are very different compared to software engineering patterns. However, model transformation as a research area can provide us with useful concepts and patterns that will help in the definition of our theoretical framework in order to address the issue of maintaining consistency between workflow models at different levels of abstraction.

2.3 Workflow Process Modelling and Execution

The aim of our research is to propose a framework that could easily fit and work with popular and widely adopted approaches and not to generate 'yet another method' that will stay only in theory and have no real adoption by industry. Thus, among numerous research proposals for workflow process languages and process execution frameworks we review those that fulfil the following criteria:

1. Industry adoption; popularity and adoption by software houses and vendors, IT society.
2. Standardization efforts; adoption by standardization organizations such as WfMC [7], BPMI [74], OASIS [75], OMG [55], and NIST [76].
3. Referent literature; related research comparing languages and execution frameworks [77], [78], [79], [4], [80], [81], [2].

The remaining of this Section introduces the main concepts of each selected language and framework and identifies the main limitations through the literature.

2.3.1 Workflow Process Modelling Languages

The term *Workflow Process Modelling Languages* includes all process modelling methods and techniques used for capturing high-level representations of business processes within the workflow context. This kind of languages usually offer a number of visual constructs that IT designers use to model business behaviour. Thus, providing visual representations of processes they help communication across business and IT people [52]. The workflow process modelling languages that we review are:

- Integration Definition for Function Modelling [IDEF0] [36],
- Event Driven Process Chain [EPC] [35],
- Integration Definition for Process Description [IDEF3] [37],
- Business Process Modelling Notation [BPMN] [38],
- XML Process Definition Language [XPDL] [39],
- Business Process Definition Metamodel [BPDM] [82] and
- UML 2 Activity Diagrams [UML 2 ADs] [40].

Following Sections present them based on the chronological order of their proposal.

2.3.1.1 1981 - Integration Definition for Function Modelling [IDEF0]

In 1970s US Air Force initiated a program for Integrated Computer Aided Manufacturing (ICAM) [36]. The purpose of the program was to increase productivity of manufacturing through systematic application of computer technology. A series of modelling methodologies, known as the ICAM Definition (IDEF) methods, were developed by that program in order to provide better analysis and communication techniques for people involved in improving manufacturing productivity [36]. Among the IDEF methods developed, there was IDEF0. Later, in 1993, Institute of Electrical and Electronics Engineers (IEEE) Computer Society [83] initiated a project to establish IDEF standards across both industry and government within the standards framework of the American National Standards Institute (ANSI) [84]. The result of that effort was the IEEE Std. 1320.1-1998 for IDEF0 function modelling [85].

IDEF0 modelling technique produces function models; graphical structured representations of the functions within a system or subject area [36]. Therefore, an IDEF0 model describes what a system does, what controls it, what things it works on, what means it uses to perform its functions, and what it produces. As for the structure, the model is composed of a hierarchical series of diagrams that gradually introduce increasing levels of detail to describe functions and their interfaces within the context of a system [85].

As a function modelling language, IDEF0 has the following characteristics [85]:

1. The models are expressive and comprehensive. They are capable of graphically representing a wide variety of business operations to any level of detail.
2. It is a coherent and simple language, allowing accurate expressions and promoting consistency of usage and interpretation.
3. The models enhance the communication among analysts, architects, developers, managers, as they are easy to learn and hierarchically expose details of systems.
4. It is a well-tested technique through many years of use by the US Air Force and other government agencies and by private industry.
5. Several commercial products support development and analysis of IDEF0 models.

The basic constructs of IDEF0 models are **Boxes**, which represent functions and **Arrows**. Depending on how an arrow enters or leaves a box, we have [86]:

- **Inputs**, which are encoded with those arrows entering on the left side of a box. Inputs are consumed by a function to produce outputs.
- **Outputs**, which are encoded with those arrows leaving a box on the right side. Outputs are the data or objects produced by the function.

- **Controls**, which are encoded with arrows entering a box on the top. Controls specify the conditions required by the function to produce correct outputs.
- **Mechanisms**, which are encoded with arrows entering a box to the bottom side. Mechanisms identify the means that support the execution of the function.
- **Call arrows**, which are encoded with arrows leaving a box to the bottom side. Call arrows enable the sharing of detail between models (linking them together) or between portions of the same model.

Figure 2.1 illustrates the core elements of an IDEF0 model.

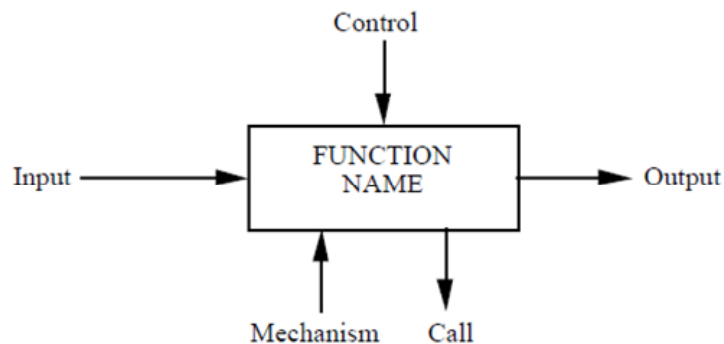


Figure 2.1: Core elements of an IDEF0 model [86]

Example:

Figure 2.2 shows a simple IDEF0 model for a computer assembly activity.

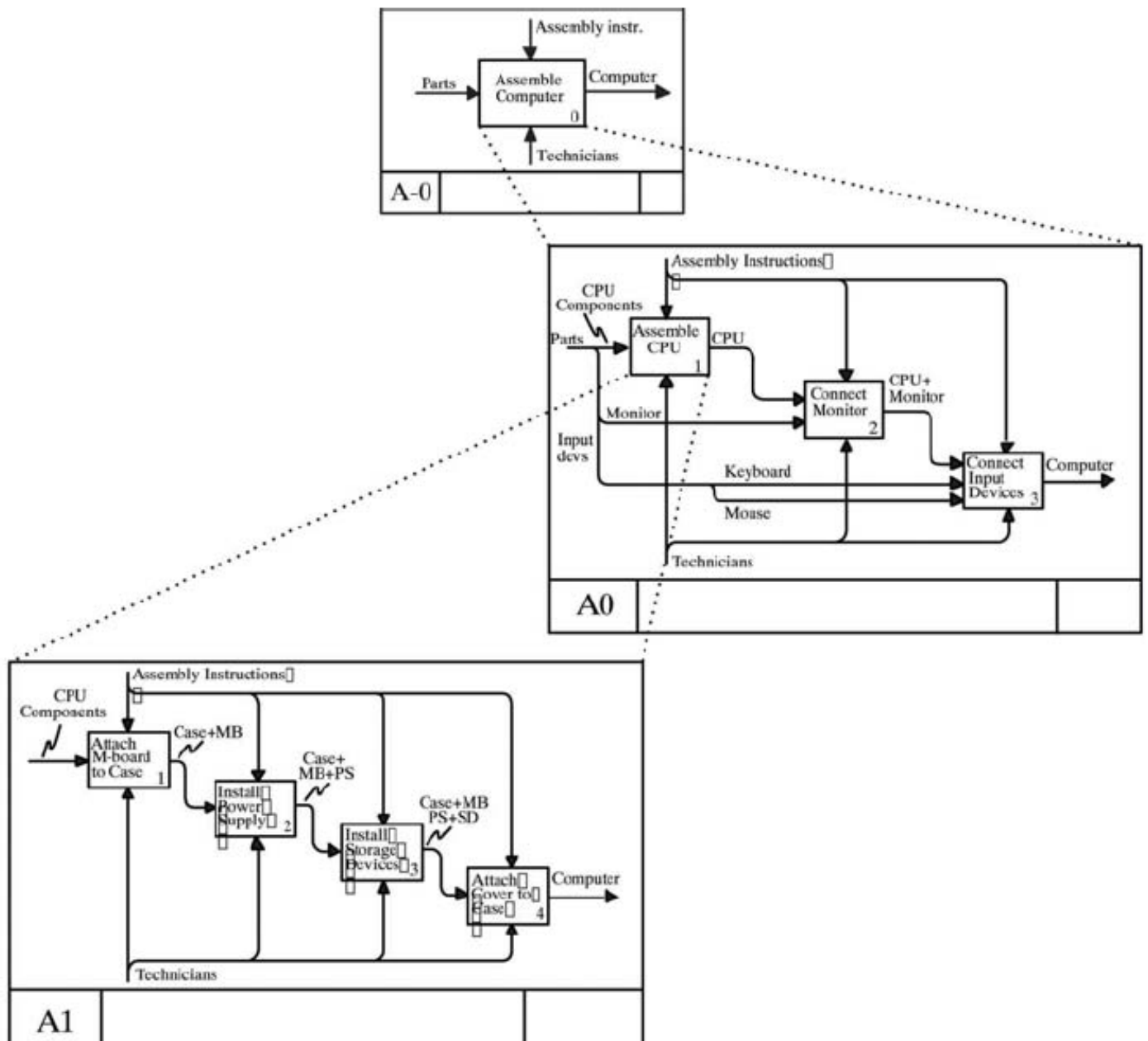


Figure 2.2: A simple IDEF0 model for a computer assembly activity [87]

Identified drawbacks:

- IDEF0 models are static diagrams with no explicit or even implicit representation of time-order constraints between activities [79], [88]. Thus, although IDEF0 allows the description of what an organization does, it does not let modellers consider timing associated with activities [88].
- IDEF0 models cannot represent behavioural or informational modelling perspectives [79]. Thus, modellers cannot describe the specific logic associated with activities [88].
- Lastly, IDEF0 modelling technique makes no assumptions about the implementation of a process, i.e. execution logic [82].

2.3.1.2 1992 - Event Driven Process Chain [EPC]

Event-driven process chain (EPC) method has been developed within the ARIS framework (Architecture of Integrated Information System) [89] and it is used by many companies for modelling, analyzing, and redesigning business processes [90]. An EPC model is an ordered graph that consists of events and functions. As a modelling method, EPC provides a variety of connectors for parallel and alternative execution of processes [91] and also supports logical operators, such as OR, AND, and XOR. One of the key advantages of EPC is its simplicity and easy-to-understand notation [92].

The main elements of an EPC are [93]:

1. Event (represented as hexagon)

Events are passive elements that describe the conditions for a function or a process to work or the state that a function or a process results.

2. Function (represented as rounded rectangle)

Functions are active elements, modelling the tasks or activities of a process. They essentially describe transformations from an initial state to a resulting state.

3. Organization unit (represented as an ellipse with a vertical line)

Organization units are used to define the roles within a process that are responsible for specific functions.

4. Information, material, or resource object (represented as rectangle)

These objects represent objects of the real world, such as documents, entities, etc., which can be input or output data of functions.

5. Logical connector

Logical connectors describe the logical relationships between elements (events and functions) in the control flow. There are three kinds of logical relationships:

1. **Branch/Merge:** Used for deciding which path to choose among several control flows.
2. **Fork/Join:** Used for activating all paths in the control flow concurrently.
3. **OR:** Used for activating one or more paths among control flows.

6. Process path (represented as rounded rectangle in front of a hexagon)

Process paths show the connection from or to other processes.

7. Control flow (represented as a dashed arrow)

It connects events with logical connectors, process paths, or functions, creating a sequence and a logical association between them.

8. Information flow

Information flows point the connection between the functions and their input or output data.

Example:

Figure 2.3 shows an EPC model, capturing the processing of a customer order.

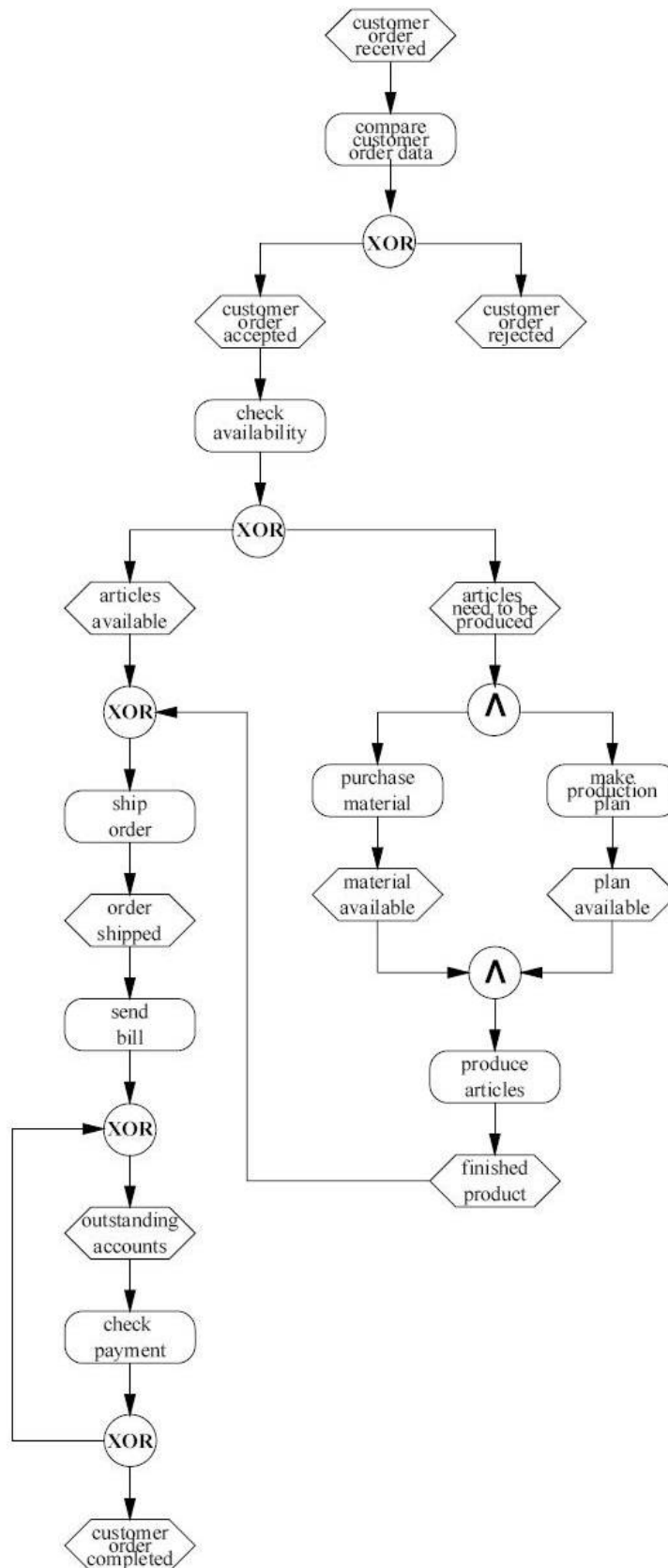


Figure 2.3: Modelling of a business process, using EPC [94]

Identified drawbacks of EPCs:

- Although there are some formalization efforts [94], overall, there are no well-defined semantics and syntax of EPCs. As a result, such models may be ambiguous. In addition, it is impossible to check these models for consistency and completeness. So it becomes obvious that it is not a safe option to use EPCs for specification of business processes that may be processed by ERP and Workflow Management systems [95].
- The fact that EPCs have no formal semantics, prevents the use of analytical techniques in these models and makes difficult the exchange of such models between tools of different vendors [95].
- EPCs do not support detailed specification of the conditions, of the control flow, of the data flow, and of the functions of a process model [96].
- Overall, EPCs are suitable only for high-level, visual models of business processes.

2.3.1.3 1993 - Integrated Definition for Process Description [IDEF3]

IDEF3, officially named as Integrated Definition for Process Description Capture Method, is considered complementary to IDEF0. IDEF3 was developed to describe behavioural aspects of processes, providing a structured method for expressing knowledge about how a process works, what precedence, and causality relations exist between activities and what their sequence is. The resulting IDEF3 descriptions provide a structured knowledge base for constructing analytical and design models [97].

IDEF3 has two model entities that form the basic units of an IDEF3 description [97]:

1. **Process flow descriptions**, describing the relationships between actions within the context of a process. The graphical elements of these models include Unit of Behaviour (UOB) boxes, precedence links, junctions, referents and notes [37]:
 - o UOB: represent elements as boxes in IDEF3 descriptions
 - o Links: connect UOB boxes to form representations of dynamic processes.
 - o Precedence Links: express temporal precedence relations between UOBs.
 - o Junctions: provide a mechanism to specify the logic of process branching.
 - o Referents: enhance understanding and provide additional meaning.
2. **Object state transition networks (OSTN)**, describing the allowable states of objects throughout a process. The basic elements of OSTNs are objects and state transition arcs [97]:
 - o Object states: represented by circles
 - o State transition arcs: represented by lines connecting object states

Example:

Figure 2.4 and Figure 2.5 show an IDEF3 process description diagram and an object state transition network diagram respectively.

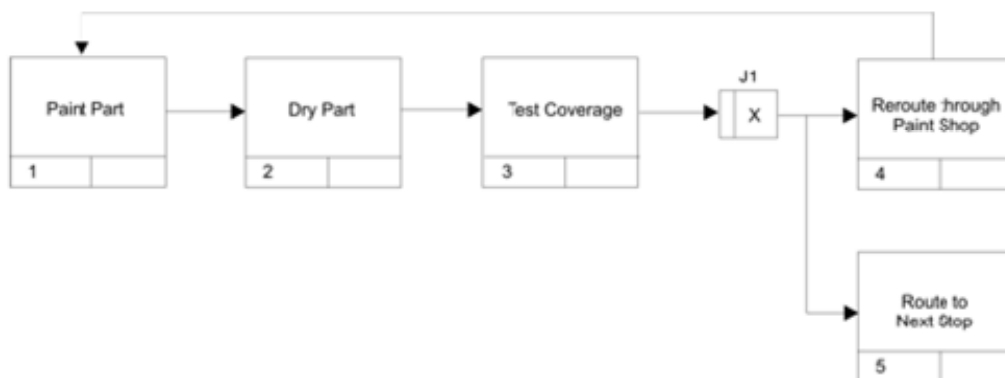


Figure 2.4: IDEF3 process description diagram [97]

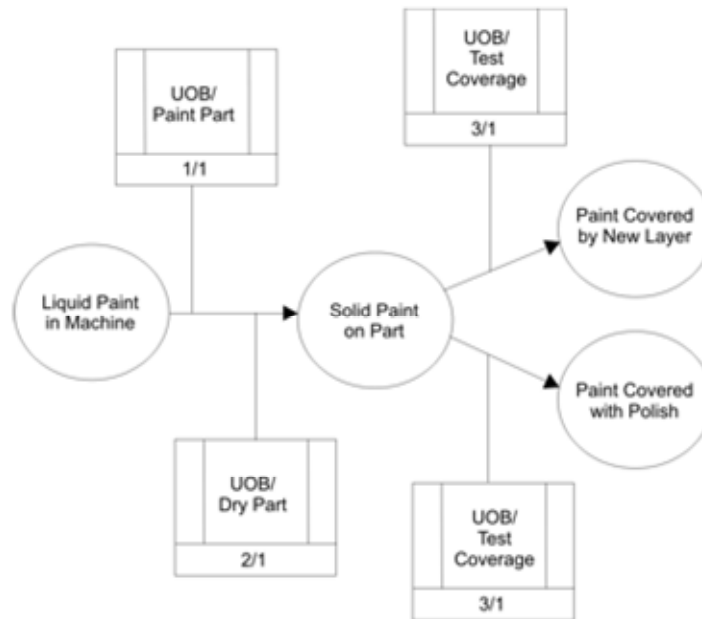


Figure 2.5: IDEF3 Object State Transition Network Diagram [97]

Identified drawbacks:

- IDEF3 lacks a mathematical foundation, not supporting rigorous modelling principles that underlie the development of complex models [98], [99]. Specifically, it is lacking of timing information that goes beyond UOB sequencing [99], has limited support for capturing dynamic behaviours, and weakly supports representation and manipulation of objects and their joint state-space [99].
- IDEF3 is neither appropriate for handling process change, as it offers limited capabilities to portray the organisational and behavioural perspectives [100], nor able to represent flow of multiple objects along the single link that connects a pair of UOBs.
- Another disadvantage of this methodology is that it produces an elaborate net of models in order to achieve model integration. In cases where the net of models is too complex, it would almost be impossible for a user to follow and understand it [100].
- Overall, IDEF3 is considered as a semiformal knowledge capture approach [99].

2.3.1.4 2002 - Business Process Modelling Notation [BPMN]

BPMN was initially developed by Business Process Management Initiative (BPMI), and is currently maintained by the Object Management Group [101]. The purpose of BPMN is to enable businesses to understand their internal procedures with a graphical notation and give them the capability to communicate these procedures in a standard manner. Furthermore, BPMN models facilitate the understanding of performance collaborations and business transactions between the organizations. The ultimate goal is that using BPMN, businesses will understand themselves and their collaborators and this knowledge will enable quick adjustment to new internal and business-to-business circumstances [38].

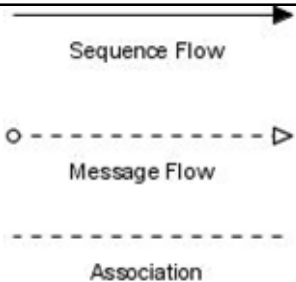
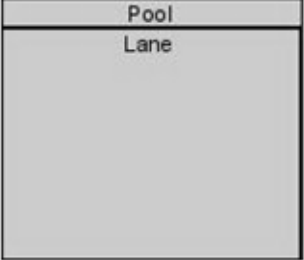
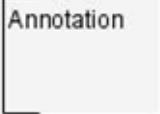
 Start Intermediate End	Event
 Task Process	Activity
 Gateway Fork/Join Inclusive Decision/Merge	Gateway
 Sequence Flow Message Flow Association	Connections
 Pool Lane	Swimlanes
 Group	Groups
 Annotation	Annotation

Figure 2.6: Graphical elements of BPDs [101]

BPMN defines a Business Process Diagram (BPD), which, based on a modified flowcharting technique, creates models of business process operations in a graphical manner. BPDs are essentially networks of graphical objects, comprising of activities and flow controls that define the order of performance [52]. A BPD consists of a set of graphical elements that enable easy development of simple diagrams and look familiar to most business analysts [52]. The four basic categories of elements are [101]:

1. **Flow Objects:** Events, Activities, Gateways
2. **Connecting Objects:** Sequence Flow, Message Flow, Association
3. **Swimlanes:** Pool, Lane
4. **Artefacts:** Data Object, Group, Annotation

Figure 2.6 above shows the basic elements of BPDs.

As for the structure types of processes, there are three basic types [78]:

- **Private (Internal) business process;** define internal business process of an organization and concerns the workflow definitions in general.
- **Abstract (Public) business process;** define the interaction between a private business process and other processes or participants.
- **Collaboration (Global) business process;** define the interactions between two or more business entities.

Example:

Figure 2.7 shows an example model of a travel booking process in BPMN.

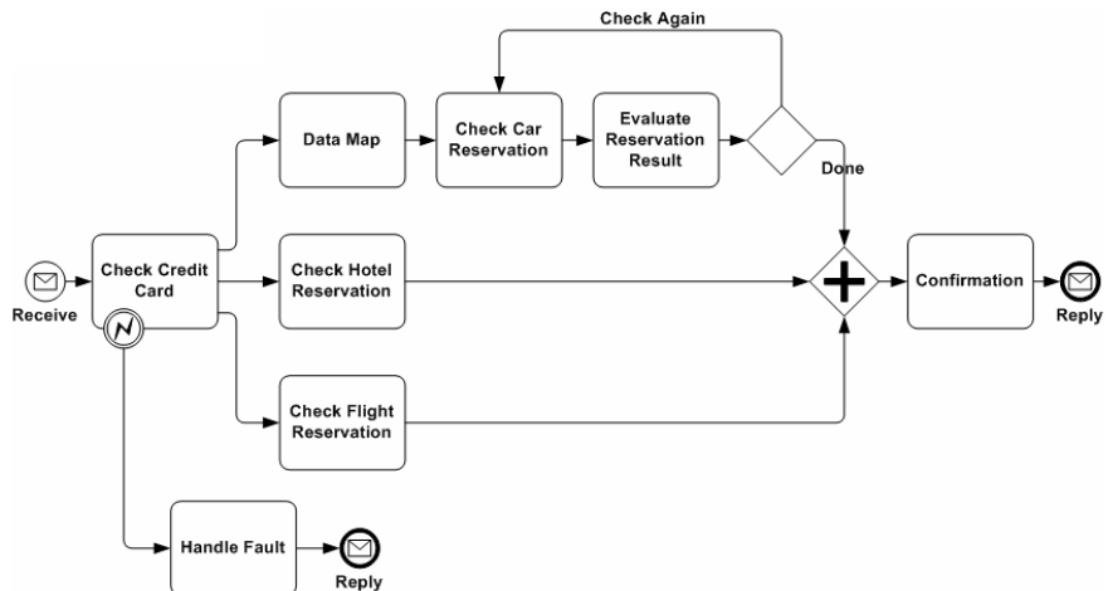


Figure 2.7: A travel booking process modelled with BPMN [102]

Identified drawbacks:

- BPMN does not provide a complete technical model. Specifically, it is not possible to describe the data manipulated by the process, the data transformations, and the data results for each activity [103]. Another data related limitation is that data interaction with external sources, operated outside the context of the workflow engine is not supported [78].
- As far as the execution semantics are concerned, modellers cannot realize the workflow pattern [104] dealing with multiple instances without synchronization, as BPMN does not support spawning multiple instances of a process in an unsynchronized way [78].
- Moreover, BPMN is not extensible to define organizational structures, functional breakdowns, data and informational models and business rules [78].
- Finally, business analysts require a significant amount of training and access to technical resources in order to create a BPMN diagram properly. BPMN has many concepts and ways of constraining a process model, and it takes time to learn all the concepts and be able to turn an idea into a process model [103].

2.3.1.5 2002 - XML Process Definition Language [XPDL]

The XML Process Definition Language XPDL was designed by the Workflow Management Coalition (WfMC) [7], an organization with members representing all facets of workflow; from vendors to users, and from academics to consultants [105]. The goal of XPDL was to provide a universally accepted process design format for storing and exchanging process diagrams, so that one tool would model a process diagram, another would read and edit it, an XPDL-compliant process engine would execute the process model, and so on [106].

XPDL uses an XML-based syntax, specified by an XML schema. The main elements of the language are [107]:

- **Package:** This container holds the other elements of the process diagram.
- **Application:** This container specifies the applications and tools invoked by the workflow processes defined in a package.
- **Workflow-Process:** It defines parts or whole workflow processes and is composed of elements of type Activity and Transition.
- **Activity:** This is the basic building block of a workflow process definition.
- **Transition:** Elements of this type connect elements of type Activity.
- **Participant:** This container specifies the participants of the workflow process, i.e., the entities that can execute work.
- **DataField** and **DataType:** These containers specify workflow relevant data.

Example:

As XPDL uses an XML-based syntax, we show in Figure 2.8 the respective “grid view” of a sample workflow process [108] as captured in the Altova XML Spy 2007 editor [109] in order to avoid showing complex xml listings.

XML

Package

Id

wfmc-sample

Name

Sample Workflow Process

xmlns

http://www.wfmc.org/2002/XPDL1.0

xmlns:obe

http://obe.sourceforge.net/2005/OBE1.0

xmlns:xpdl

http://www.wfmc.org/2002/XPDL1.0

xmlns:xsi

http://www.w3.org/2001/XMLSchema-instance

xmlns:xyz

http://www.xyzeorder.com/workflow

xsi:schemaLoca...

http://www.wfmc.org/2002/XPDL1.0 http://wfmc.org/standards/docs/TC-1025_schema_10_xpdl.xsd

PackageHeader

ConformanceClass

GraphConformance=NON_BLOCKED

Script

Type=text/x-xpath

TypeDeclarations

Participants

Applications

WorkflowProcesses

WorkflowProcess (3)

AccessLevel

Id

Name

ProcessHeader

1

PUBLIC

wfmc-wf-1

EOrder

2

PRIVATE

wfmc-wf-2

FillOrder

3

PRIVATE

wfmc-wf-3

CreditCheck

ExtendedAttributes

Figure 2.8: An XPDL sample workflow process

Identified drawbacks:

- XPDL does not have formal semantics [77] and does not support any transaction or exception semantics. Specifically, there is no explicit transaction demarcation and as for exceptions, there is no notion of fault or compensation handler [110, 111].
- XPDL has limited support for modelling flow of activities inside a process [107].
- Overall, as XPDL was not primarily designed for modelling, it has no graphical notation, it is hard to read and requires technical background on XML language [105].

2.3.1.6 2003 - Business Process Definition Metamodel [BPDM]

Business Process Definition Metamodel (BPDM) is the result of an open process involving submissions by organizations, following a Request for Proposal issued in 2003. BPDM was adopted in initial form in 2007, and finalized in 2008 [112]. The expectations for BPDM were to become a common metamodel that would [113]:

- unify the different business process definition graphical and textual notations existing in the industry.
- complement UML metamodels so that business processes specifications could be part of complete system specifications, assuring consistency and completeness.
- integrate collaborations between business units, process models for workflow management processes and automated business processes.
- support the specification of choreography, describing the collaboration between participating business entities.
- enable the exchange of business process specifications between modelling tools, and between tools and execution environments using XMI (XML Metadata Interchange) [114].

The basis of BPDM's definition is the notion of meta-model. A meta-model is a model of how to describe business processes; it is a kind of shared vocabulary of process with well-defined connections between terms and concepts [115]. In order to integrate the notations and technologies and leverage existing assets and new designs, the meta-model captures their meaning. The meta-model behind BPDM uses the OMG "Meta Object Facility" (MOF) standard [116] to capture business processes and provide an XML syntax for storing and exchanging them between tools. Various tools, methods, and technologies can then map their way to view, understand, and implement processes to and through BPDM.

BPDM supports two views of process [117]:

- **Orchestration view:** This view describes what happens and when. Concepts are represented through sequences of Activities that produce results with branching and synchronization. Orchestration is typically represented as flow charts, activity diagrams, swim lanes or similar notations of one task or activity following another.
- **Choreography view:** This view describes how semi-independent and collaborating entities work together in a process. Choreography captures the interactions of roles with well-defined responsibilities within a given process

Choreography and orchestration are effectively two sides of a business process model and BPDM tries to join them into a unified model [117].

The basic notation elements of BPDM are shown in Figure 2.9.

Element	Notation
Service Specification	
Service Provider	
Process	
Tasks	
Flow (depending on the type of line and the starting and ending points it denotes control flow, data flow, exception flow and message flow)	
Data Store (stores information between process invocations)	
Note (conveys additional comments on a diagram)	

Figure 2.9: Basic notation elements of BPDM [118]

Example:

Figure 2.10 shows a part of a banking process model using BPDM.

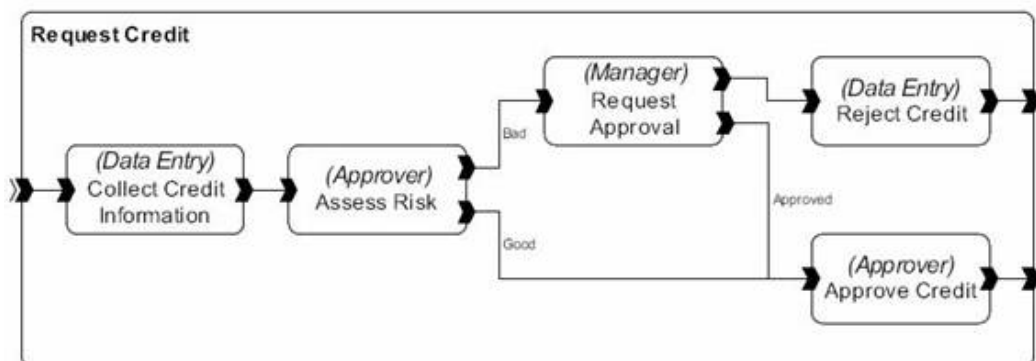


Figure 2.10: An example of a process model using BPDM [118]

Identified drawbacks:

- Industry and organisations has not yet adopted BPDM. There is no tool up to date supporting the specification of BPDM [105].
- BPDM deals with descriptions of business processes at a level abstracted from any implementation technology, making no assumptions about the form of the execution [105].
- Business processes often require information about an organisation in order to determine appropriate routing for an action, such as an approval or selection of performers. Nevertheless, the organisational model is out of scope for BPDM definition [105].
- Finally, BPDM does not provide explicit integration of business rules [105].

2.3.1.7 2004 - UML 2 Activity Diagrams [UML 2 ADs]

In 1996, a company named Rational Software began the development of a non-proprietary Unified Modelling Language. An international consortium, called the UML Partners, was organized that year to complete the Unified Modelling Language (UML) specification and the result of this work, UML 1.1, was submitted and adopted by the Object Management Group (OMG) [55] in 1997 [119]. UML 2 is the latest version of UML. UML 2 Activity Diagrams is a subset of UML 2, which defines the notation and semantics for diagrams that represent dynamic behaviour of systems [55]. UML 2 Activity Diagrams (UML 2 ADs) are typically used for business process modelling to model the workflow behind the system being designed [120], describing what actions need to take place and when they should occur [121].

UML 2 ADs are read from top to bottom, showing the flow of activities through the system. Branches and forks in the diagrams describe conditions and parallel activities respectively [120]. A UML 2 AD consists of the following behavioural elements [122]:

- **Activity:** It is the basic element and represented by a rectangle with rounded edges.
- **Initial Activity:** It is the starting point or first activity of the flow. It is represented by a solid circle.
- **Decision:** Logical condition based on which a choice has to be made. It is represented by a diamond.
- **Signal:** An Activity is called a Signal when it sends or receives a message. There are two types of signals: (i) Input signals, which are Activities that receive messages and are represented by concave polygons and (ii) Output signals, which are Activities that send messages and are represented by convex polygons.
- **Concurrent Activities:** Those Activities occur simultaneously or in parallel. Concurrency is represented by a horizontal split and the concurrent activities next to each other.
- **Final Activity:** It is the ending point of the flow. It is represented by a bull's eye symbol.
- **Swimlane:** It is a way to group activities.

Example:

Figure 2.11 shows a simple process of an order request.

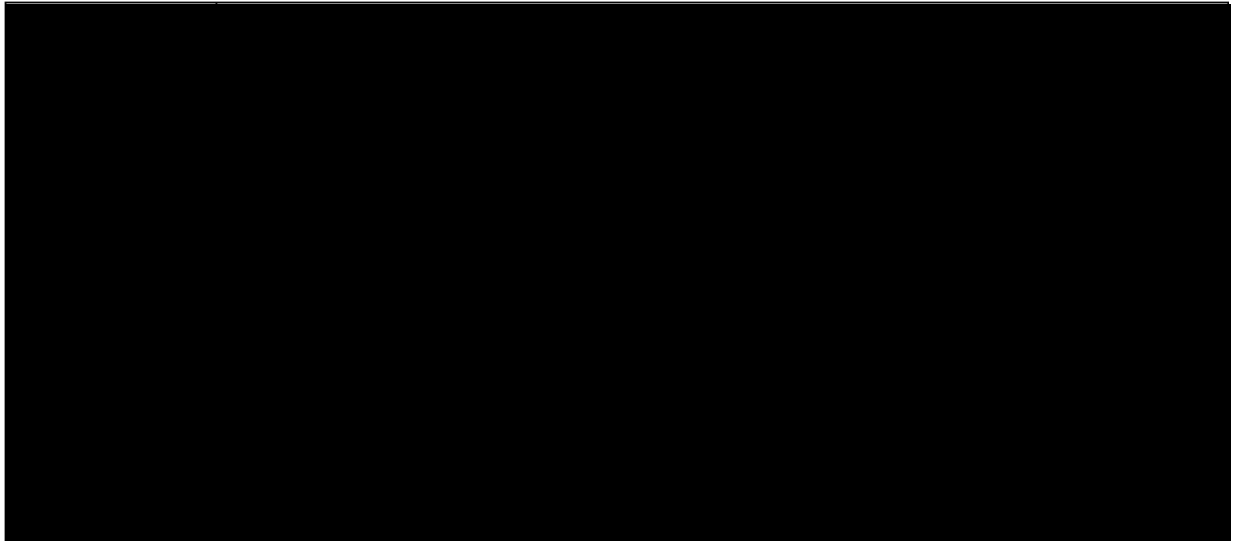


Figure 2.11: An Order Request modelled with UML 2 AD [33]

Identified drawbacks:

- UML 2 ADs do not support modelling resource-related aspects of business processes. Specifically, there is no modelling constructs in order to capture utilisation of data resources either within the model or externally from the environment [123].
- Apart from resource-related aspects, UML 2 ADs have limited support for modelling organisational aspects of business processes, such as the notion of interaction with the operational environment in which the process functions and the work distribution. Specifically, UML 2 ADs have limited support for modelling any form of work distribution other than direct allocation or role-based allocation [123].

2.3.2 Workflow Process Execution Frameworks

Workflow Process Execution Frameworks focus on the definition of low-level workflow execution semantics that will be eventually executed by an execution engine. The industry has taken a step forward in the direction of distributed workflow execution modelling and has introduced a number of execution languages, such as WSFL [41], XLANG [42] and most lately BPEL4WS [43]. These languages have adequate programming capabilities, such as variable declarations, control structures, and fault handlers. They also provide notations for describing interactions of web-services as business processes. At the same time, Semantic Web community has developed powerful representation and reasoning technology [124-126] in the direction of semantic representation, but has remained largely disconnected from the industrial effort [127]. Semantic Web Community has introduced OWL-S as an ontology-based language for describing services [44]. The ontology enables the definition of services content vocabulary in terms of objects and complex relationships between them, including class, subclass, and cardinality restrictions. Thus, there are mainly two categories of execution frameworks; mathematical based and ontology based, as shown in Table 2.1.

Mathematical Based	Ontology Based
• 2001 - Application of Petri Nets with Web-services Flow Language WSFL (IBM) [41] • 2001 - Pi-Calculus model with XLANG (Microsoft) [42] • 2002 - Business Process Execution Language for Web-services BPEL4WS (BEA, IBM, and Microsoft) [43]	• 2003 - Ontology Web Language OWL-S (<i>formerly DAML-S [125]</i>) (DAML Researchers) [44] • 2005 – Web Service Modelling eXecution environment WSMX (WSMX Working Group) [45]

Table 2.1: Categorization of Process Execution Frameworks

The main characteristic of mathematical based execution frameworks is that they are prescriptive. The expressiveness of business process behaviour is somewhat constrained, as in most of the cases it has to comply with mathematical models and relations (Petri Nets and Pi-calculus) and with rudimentary content languages (XML and XML schemas). In the mathematical based category we include WSFL [41], XLANG [42] and BPEL4WS [43]. BPEL4WS is considered as the representative framework of the category. BPEL4WS is a specification co-authored by IBM [128], Microsoft [129], BEA [130], SAP [131], and Siebel Systems [132] and merges ideas from Microsoft's XLANG and IBM's WSFL. On the other hand, although ontology based execution framework have some mathematical foundation [133], they are mainly semantic oriented. They aim to make business processes computer-interpretable, described with sufficient information in order to enable "automatic" business process discovery, invocation, composition, and execution monitoring. In the ontology based category we include OWL-S [44] and WSMX [45], as two parallel efforts of semantic web community [53] and consider OWL-S as the representative framework of the category.

2.3.2.1 2002 - Business Process Execution Language for Web Services[BPEL4WS]

BPEL4WS specification was co-authored by IBM [128], Microsoft [129], BEA [130], SAP [131], and Siebel Systems [132] and merges ideas from Microsoft's XLANG [134] and IBM's WSFL [135], [43]. It provides a notation for describing interactions of Web-services as business processes [127], using an XML-based language. Workflows in BPEL4WS are directed by traditional control structures; if, then, else, and while-loop. Services are integrated by treating them as *partners* that fill *roles* in a BPEL4WS *process model*, while Web-service Description Language (WSDL) [23] documents describe the communication-level parameters of the partner services.

BPEL4WS process model describes a program that orchestrates the interaction of the service partners. The key components of a BPEL4WS process model are [127]:

- **partners**, associating a Web-service with a particular role
- **variables**, containing the messages passed between partners and corresponding to messages in accompanying WSDL documents
- **fault handlers**, dealing with exceptions
- **flow**, which lists the *activities* defining the control flow of the process.

The basic structure of BPEL4WS language is shown in Figure 2.12:

```
<process name="ncname" targetNamespace="uri"
  queryLanguage="anyURI"?
  expressionLanguage="anyURI"?
  suppressJoinFailure="yes|no"?
  enableInstanceCompensation="yes|no"?
  abstractProcess="yes|no"?
  xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/">

  <partnerLinks>?
    <!-- Note: At least one role must be specified. -->
    <partnerLink name="ncname" partnerLinkType="qname"
      myRole="ncname"? partnerRole="ncname"?>+
    </partnerLink>
  </partnerLinks>
  <partners>?
    <partner name="ncname">+
      <partnerLink name="ncname"/>+
    </partner>
  </partners>
  <variables>?
    <variable name="ncname" messageType="qname"?
      type="qname"? element="qname"?/>+
  </variables>
  <correlationSets>?
    <correlationSet name="ncname" properties="qname-list"/>+
  </correlationSets>

  <faultHandlers>?
    <!-- Note: There must be at least one fault handler or de-
      fault. -->
    <catch faultName="qname"? faultVariable="ncname"?>*
```

```

        activity
      </catch>
      <catchAll>?
        activity
      </catchAll>
    </faultHandlers>
    <compensationHandler>?
      activity
    </compensationHandler>
    <eventHandlers>?
      <!-- Note: There must be at least one onMessage or onAlarm
      handler. -->
      <onMessage partnerLink="ncname" portType="qname"
        operation="ncname" variable="ncname"?>
        <correlations>?
          <correlation set="ncname" initiate="yes|no"?>+
        </correlations>
        activity
      </onMessage>
      <onAlarm for="duration-expr"? until="deadline-expr"?>*
        activity
      </onAlarm>
    </eventHandlers>
  activity
</process>

```

Figure 2.12: Basic structure of BPEL4WS language [43]

Example:

Figure 2.13 shows an example of a BPEL4WS process where a seller and a buyer are negotiating prices.

```

<?xml version="1.0" encoding="UTF-8"?>
<process xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:tns="http://docs.active-
endpoints.com/activebpel/sample/wsdL/marketplace/2006/09/marketplace.wsdl"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="marketplace" targetName-
space="http://docs.active-
endpoints.com/activebpel/sample/wsdL/marketplace/2006/09/marketplace.wsdl">
  <import importType="http://schemas.xmlsoap.org/wsdl/" loca-
tion="project:/BPEL_Samples/Resources/WSDL/marketplace.wsdl" name-
space="http://docs.active-
endpoints.com/activebpel/sample/wsdL/marketplace/2006/09/marketplace.wsdl"/>
  <partnerLinks>
    <partnerLink myRole="sales" name="seller" partnerLink-
Type="tns:salesplnk"/>
    <partnerLink myRole="buying" name="buyer" partnerLink-
Type="tns:buyingplnk"/>
  </partnerLinks>
  <variables>
    <variable messageType="tns:sellerInfoMessage" name="sellerInfo"/>
    <variable messageType="tns:negotiationMessage"
name="negotiationOutcome"/>
    <variable messageType="tns:buyerInfoMessage" name="buyerInfo"/>
  </variables>
  <correlationSets>
    <correlationSet name="negotiationIdentifier" proper-
ties="tns:negotiatedItem"/>
  </correlationSets>
  <sequence name="MarketplaceSequence">
    <flow name="MarketplaceFlow">
      <receive createInstance="yes" name="SellerReceive" opera-
tion="submit" partnerLink="seller" portType="tns:sellerPT" vari-
able="sellerInfo">
        <correlations>
          <correlation initiate="join" set="negotiationIdentifier"/>

```

```

        </correlations>
    </receive>
    <!-- tets -->
    <receive createInstance="yes" name="BuyerReceive" operation="submit" partnerLink="buyer" portType="tns:buyerPT" variable="buyerInfo">
        <correlations>
            <correlation initiate="join" set="negotiationIdentifier"/>
        </correlations>
    </receive>
</flow>
<if name="MarketplaceSwitch">
    <condition>($sellerInfo.askingPrice <= $buyerInfo.offer)</condition>
    <assign name="SuccessAssign">
        <copy>
            <from>'Deal Successful'</from>
            <to part="outcome" variable="negotiationOutcome"/>
        </copy>
    </assign>
    <else>
        <assign name="FailedAssign">
            <copy>
                <from>'Deal Failed'</from>
                <to part="outcome" variable="negotiationOutcome"/>
            </copy>
        </assign>
    </else>
</if>
    <reply name="SellerReply" operation="submit" partnerLink="seller" portType="tns:sellerPT" variable="negotiationOutcome"/>
    <reply name="BuyerReply" operation="submit" partnerLink="buyer" portType="tns:buyerPT" variable="negotiationOutcome"/>
</sequence>
</process>

```

Figure 2.13: An example of a process in BPEL4WS [136]

Identified drawbacks:

- BPEL4WS neither defines a graphical representation of processes nor provides any particular design methodology for processes [137].
- BPEL4WS relies on XML for describing services. XML provides a rudimentary content language, but lacks the constructs to describe complex relationships between Web resources [127].
- BPEL4WS cannot express the inheritance and relationships among the web services inside a process [138].
- BPEL4WS has the drawback of statically binding the existing services, referencing fixed WSDL files. As a result, it regards the binding relationship between processes and services as a known condition and makes the workflow less flexible in dynamic and ubiquitous environments [139], [140].

2.3.2.2 2003 - OWL-S

The lack of semantics in the industry backed web-services standards, led the Semantic Web Community to develop a DAML+OIL ontology for Web-services known as OWL-S (2003) [44], formerly DAML-S [125]. OWL-S is an ontology for describing web-services based on DAML+OIL [141]. As a DAML+OIL ontology, OWL-S has well-defined semantics, making it computer-interpretable and unambiguous. It also enables the definition of Web-services content vocabulary in terms of objects and complex relationships between them, including class, subclass, and cardinality restrictions. The OWL-S upper ontology comprises three components [44], [127]:

1. **ServiceProfile:** It describes the properties of a service necessary for automatic discovery, such as what the service offers, its inputs and outputs, its preconditions and effects.
2. **ServiceModel:** It describes the process model of a service, i.e. the control flow and data flow involved in using the service.
3. **ServiceGrounding:** It connects the process model description to communication-level protocols and message descriptions in WSDL.

The unique aspects of OWL-S, in comparison with other execution frameworks are [127]:

- OWL-S can express hierarchies and taxonomic info: OWL-S classes may draw properties from inheritance and other relationships to other OWL-S classes, thus providing for a richer representation of an individual service and the relationships between services.
- OWL-S ServiceProfile and ServiceModel provide sufficient information to enable automated discovery, composition, and execution based on well-defined descriptions of a service's inputs, outputs, preconditions, effects, and process model.

Example:

Figure 2.14 shows a piece of OWL-S code, showing an example of a service definition.

```
<!-- CoastalAreaOilCleaning SERVICE -->
<service:Service rdf:ID="CoastalAreaOilCleaningService">
  <!-- Reference to the Profile -->
  <service:presents rdf:resource="#CoastalAreaOilCleaningProfile"/>
  <!-- Reference to the Process Model -->
  <service:describedBy rdf:resource="#CoastalAreaOilCleaning"/>
  <!-- Reference to the Grounding -->
  <service:supports>
    <grounding:WsdGrounding rdf:ID="EmptyGrounding"/>
  </service:supports>
</service:Service>
```

Figure 2.14: Example of a service definition in OWL-S [142]

Identified drawbacks:

- OWL-S results to large service descriptions that are difficult to read and write. Moreover, it does not support the description of certain rule types often needed for service description [143].
- Another drawback of OWL-S is that users have reported to have serious difficulties in learning it and using it [144], a fact that could limit its adoption.
- As far as the operations' semantics are concerned, OWL-S cannot expressively capture them. There are two key reasons for this: (1) OWL-S does not allow for the use of logic variables like in rule languages and (2) the logical formulas are limited to the predefined description logic of OWL-S [143]. As a result OWL-S cannot describe rules that are frequently addressed when describing processes such as "if the price is below credit card's limit, approve transaction" etc.
- Finally, there are serious limitations on a conceptual level as the formal semantics of OWL-S are not entirely clear [145]. A characteristic example is that although OWL-S offers the choice between a number languages for the specification of preconditions and effects, it is not entirely clear how these languages interact with OWL-S [146].

2.3.2.3 2005 - Web Service Modelling eXecution environment [WSMX]

WSMX (Web Service Modelling eXecution environment) is the reference implementation of WSMO (Web Service Modelling Ontology) [147]. It is an execution environment that aims to increase flexibility in business processes automation and provide scalable integration solutions [45]. The development of WSMX was initiated and driven by three major directives [45]:

1. The need to have a semantic execution environment capable of manipulating semantic messages, discovering semantically enriched web services, invoking and composing them for the end-user benefit.
2. The need to have a functional service open architecture prototype, which would be capable to encapsulate new components with new functionalities.
3. The need for reusability of component and general functionality.

Semantic Web's aim is to enable machines to automatically carry out tasks using web services with a minimum or no human intervention [148]. These tasks include automatic discovery, composition, and execution. WSMX is moving towards this goal by allowing web services whose semantics have been formally described in Web Service Modelling Language to be discovered, composed, and executed. WSMX uses Web Service Modelling Language (WSML) [146] which is based on the WSMO conceptual model and provides a Rule Language for the Semantic Web.

Example:

Figure 2.15 shows an example for a Concept and Axiom Description in WSML.

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-flight"

namespace {_"http://www.example.org/Family#",
  dc _"http://purl.org/dc/elements/1.1#"}

ontology Family
  concept Human
    hasParent inverseOf(hasChild) ofType Human
    hasChild ofType Human
    hasAgeInYears ofType (0 1) _integer

  axiom DefinitionTeenager
    nonFunctionalProperties
      dc#source hasValue _"http://dictionary.reference.com/search?q=teen"
    endNonFunctionalProperties
    definedBy
      forall {?teen, ?age} (
        ?teen memberOf Teenager impliedBy
        ?teen[hasAgeInYears hasValue ?age] memberOf Human and
        ?age >= 13 and ?age <= 19).
```

Figure 2.15: An example for a Concept and Axiom Description in WSML [149]

Identified drawbacks:

- WSMX does not implement orchestration. Moreover, complete automatic discovery of services is still not available [150].
- WSMX supports only MSML format and does not recognize other message formats [151].
- WSMX does not yet support automated service composition. However, it implements late binding as a strategy for adaptation [152].
- Finally, WSMX is still in a premature phase with very little industry adoption.

2.3.2.4 Comparing BPEL4WS and OWL-S

As BPEL4WS and OWL-S are the most prominent efforts from the industry and the semantic web community respectively, in this Section, we identify the similarities and differences of these two proposals.

As discussed in [127], OWL-S and BPEL4WS have broad and somewhat complementary objectives. OWL-S's ServiceProfile complements and extends ideas in UDDI. OWL-S's ServiceGrounding connects the application level content description of a service to communication level descriptions in WSDL. In addition, ServiceModel (Process Model) in OWL-S is closely related to the business process model in BPEL4WS. Considering these complementary objectives, Mandell and McIlraith in [127] propose a bottom-up approach to integrate semantic web technology, i.e. OWL-S, into BPEL4WS models. The goal is to achieve automating customized, dynamic binding of web-services together with interoperation through semantic translation.

Another common characteristic between BPEL4WS and OWL-S is that both provide a mechanism for describing a business process model. The difference relies on the strictness of description. BPEL4WS's reliance on describing services using XML and XML Schema prevent it from describing complex relationships between web resources and expressing rich semantic information. On the contrary, OWL-S, based on DAML+OIL, enables the representation of classes, properties, domain and range, and subclass plus super-class hierarchies. OWL-S also has well-defined semantics and the ability to define complex relationships between properties of objects in an ontology [127].

Another difference relies on the semantics representation. While in OWL-S ServiceProfile and ServiceModel provide sufficient information to enable automated discovery, composition, and execution based on well-defined description of a service's inputs, outputs, preconditions, effects, and process model, in BPEL4WS there is a lack of well-defined semantics as partners are restricted by structured XML content contained in WSDL port type definition. On the other hand, BPEL4WS has extended mechanisms for fault handling, execution monitoring, and transactions, while OWL-S does not support any of these aspects, lacking a lot in programming capabilities. Finally, none of the two directly supports query mechanism to expose the state of executing processes [127].

A number of researchers try to exploit the similarities between BPEL4WS and OWL-S, in order to achieve a mapping between the two. Specifically, in [153], authors propose a technique for the automated composition of web services described in OWL-S, which allows for the automated generation of executable processes written as BPEL4WS programs. Given a set of available services, they translate OWL-S process models into nondeterministic and partially observable state transition systems that describe the dynamic interactions with external services. Moreover, in

[154] authors use their model transformation framework, named Simple Transformer (SiTra), in order to transform OWL-S descriptions to BPEL4WS descriptions.

Finally, in [155] authors underline that the rapid growth and automation demands of e-business and grid applications require BPEL4WS to provide enhanced semantic annotations to achieve the goal of business processes automation. They argue that OWL-S is designed to represent such kind of semantic information. Based on the similarity in the conceptual model of OWL-S and BPEL4WS, they try to overcome the lack of semantics in BPEL4WS by mapping the BPEL4WS process model to the OWL-S suite of ontologies. Therefore, they introduce a mapping from the BPEL4WS process model to the complete OWL-S suite of ontologies. They present a mapping strategy and a tool named BPEL4WS2OWL-S, supporting this strategy. Figure 2.16 gives an overview of the mapping proposed in this paper.

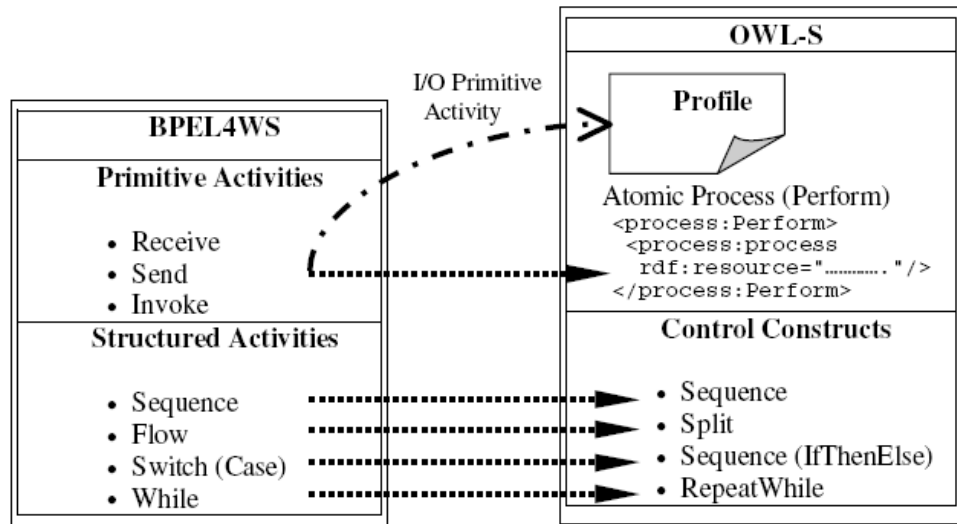


Figure 2.16: Overview of mapping BPEL4WS to OWL-S [155]

2.4 Combining High-level representations with Low-level Execution Semantics

There are some research approaches proposing mappings, in order to reduce the distance between high-level process descriptions and low-level workflow execution semantics. The proposed mappings that we review in this Section are:

1. UML to BPEL4WS mapping [46-48]
2. BPMN to BPEL4WS mapping [49-52]
3. FBPML [98] to OWL-S mapping [53]

2.4.1 UML to BPEL4WS

The basic motivation of this mapping is that UML is an OMG [55] standard, which provides a widely known visual modelling notation that is used for designing and understanding complex systems.

In [48], Gardner introduces a UML Profile, which supports modelling with a set of semantic constructs that correspond to those in BPEL4WS. Table 2.2 shows an overview of the mapping from the UML profile to BPEL4WS. The implementation of the mapping is built as an Eclipse [156] plug-in. The implemented plug-in takes as input XMI [114], the industry's standard file format for exchange of UML models, and generates BPEL4WS code along with the required WSDL and XSD artefacts.

UML Profile Construct	BPEL4WS Concept
<<process>> class	BPEL process definition
Activity graph on a <<process>> class	BPEL activity hierarchy
<<port>> associations	BPEL partner declarations
<<process>> class attributes	BPEL containers
Hierarchical structure and control flow	BPEL sequence and flow activities
Decision nodes	BPEL switch activities and transition conditions
<<receive>>,<<reply>>,<<invoke>> activities	BPEL receive, reply, invoke activities
<<protocol>> package with <<role>> classes	BPEL service links types and roles

Table 2.2: UML Profile and BPEL mapping [48]

Moreover, Mantell [46] describes a process of transforming UML models to BPEL4WS code, WSDL, and XSD files using a UML Activity Diagram. Figure 2.17 shows the diagram; boxes represent artefacts (usually files) while the ellipses represent an action or activity. The main stages of the transformation include building and exporting UML models to XMI, generating BPEL4WS,

WSDL and XSD files and finally deploying and running the process using a BPEL4WS execution engine.

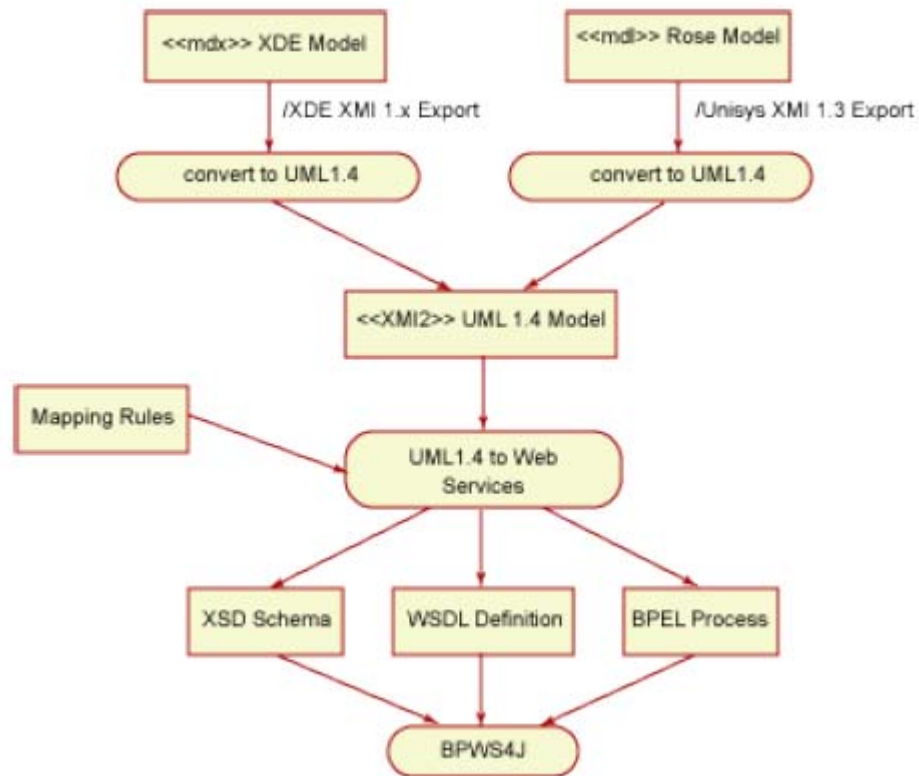


Figure 2.17: Transformation of UML model to BPEL4WS [46]

Finally, Gardner [48] discusses a number of scenarios in which UML to BPEL4WS mapping would be helpful. An interesting scenario would be to produce UML profiles in order to generate other processes implementations. Moreover, it would be possible to generate artefacts for multiple technologies based on UML models, e.g. in UML to BPEL4WS mapping, there is generation of BPEL4WS, WSDL and XSD, but non-XML outputs such as Java, or other programming language code can also be generated. Last of all, the transformation can be bi-directional, allowing the import of existing BPEL4WS and WSDL artefacts and synchronizing UML models and BPEL4WS artefacts with changes in either being reflected in the other.

2.4.2 BPMN to BPEL4WS

As White points out in his technical article [52], a key goal for the development of BPMN was to connect business-oriented process modelling notations with IT-oriented execution languages that implement the processes. Therefore, he proposes mapping the modelling structures of BPMN to BPEL4WS. Figure 2.18 shows a segment of a business process diagram in BPD with the mappings to BPEL4WS elements.

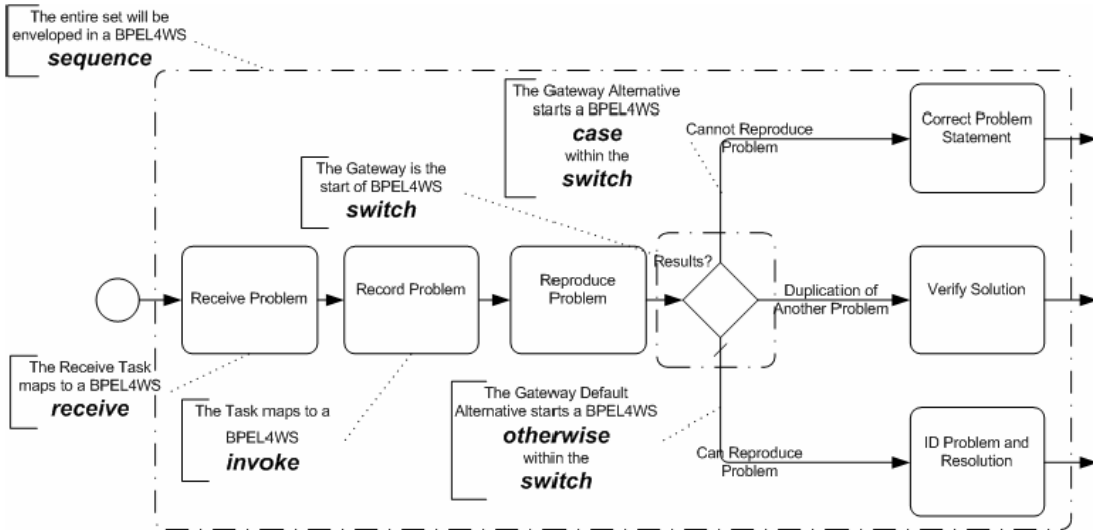


Figure 2.18: A Business Process Diagram with mappings to BPEL4WS [52]

Ouyang et al. [49, 50] argue that mapping BPMN models to BPEL4WS code is a necessary step towards unified and standards-based business process development environments and they propose a new technique focused on the control-flow perspective. To map a BPD onto BPEL4WS, BPD is decomposed into components; a component is a subset of the BPD that has one entry point and one exit point. Then the components are mapped onto suitable BPEL4WS blocks. In this way graph structures are transformed into block structures.

Based on the proposed algorithm, each component in the BPD is mapped with a BPEL4WS translation and this is repeated until no component is left in the diagram. Authors identify two categories of components for the mapping: (1) well-structured components that can be directly mapped onto BPEL4WS structured activities and (2) non-well-structured components that can be translated into BPEL4WS via event-action rules. Figure 2.19 shows examples of mappings. At the end, with this technique “patterns” in the BPMN models are discovered and BPEL4WS code is generated by discovering the mappings onto BPEL4WS block-structured constructs or acyclic graphs of control links [51].

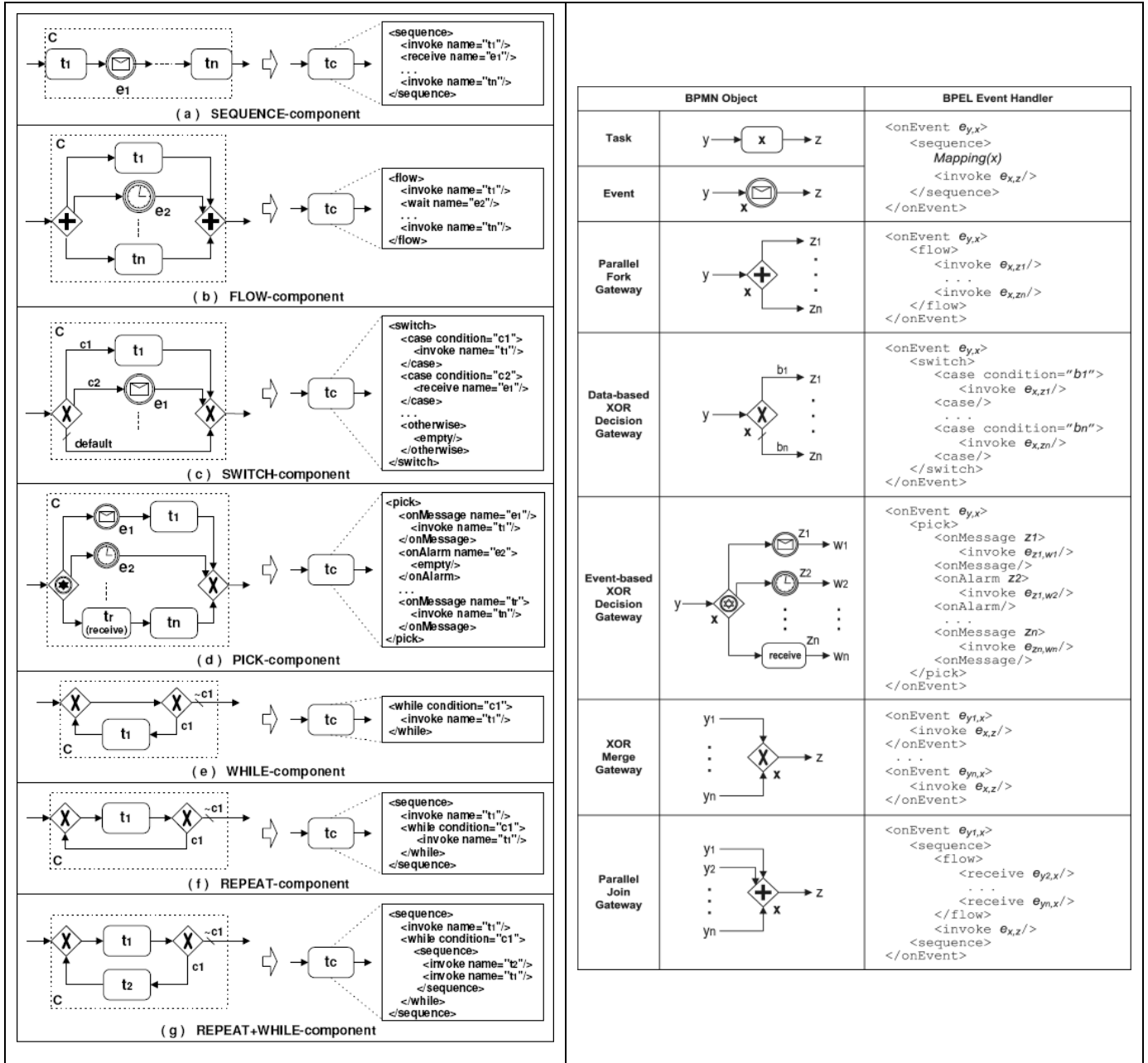


Figure 2.19: BPMN components and corresponding BPEL4WS translation [49, 50]

On the other hand, there are limitations of BPMN-to-BPEL4WS mappings because BPMN and BPEL4WS represent two fundamentally different classes of languages; BPMN is graph oriented while BPEL4WS is mainly block-structured [49]. Crucial differences between the two languages make it often impossible to generate smoothly readable BPEL4WS code from BPMN models, due to the complexity of the mapping and the compromises need to be done. Even more difficult is the problem of maintaining the BPMN model and the generated BPEL4WS code synchronized, in a way that any modification to one is correctly propagated to the other [102].

2.4.3 FBPML to OWL-S

Fundamental Business Process Modelling Language (FBPML) [98] adapts and merges two established process languages; Process Specification Language (PSL) [157], which provides the formal semantics for process modelling concepts and Integrated Definition Method IDEF3 [37], which provides visual capabilities. FBPML has two sections to provide theories and formal representations for describing data and processes [53]:

1. **Data Language**, which is first-ordered and uses the syntactic convention of Prolog [158]. It provides definitions for concepts, functions, logical qualifications, predicates, and meta-predicates.
2. **Process Language**, which is both visual and formal, providing an intuitive representation and the same convention as the Data Language.

Nadarajan et al. [53] present an ontology-based conceptual mapping framework that translates FBPML to OWL-S. Like FBPML, OWL-S has clear separation between data and process schemas:

- OWL-S's data model is described in OWL and SWRL, while FBML's is described in the FBPML Data Language.
- OWL-S contains its own classes to describe its process model, while FBPML's process model is described by the FBPML Process Language.

As a result, the mapping has also been divided into a data model part and a process model part. Figure 2.20 illustrates the mapping of classes, instances, and relationships between FBPML and OWL-S.

Concept	FBPML	OWL
Concrete Class	concrete_class(Name,Description, Example,Rules, CrossReferences, ObjectAttributes)	<owl:Class rdf:ID="ClassName"> <rdfs:comment>A com- ment</rdfs:comment> </owl:Class>
Abstract Class	abstract_class(Name,Description, Example,Rules, CrossReferences, ObjectAttributes)	<owl:Class rdf:ID="Abstract- ClassName"> <rdfs:comment>A com- ment</rdfs:comment> </owl:Class>
Instance	instance_of(InstanceName, ClassName)	<ClassName rdf:ID="Instance Name"/>

Relation- ship	FBPML	OWL
Class-to- Class Relation- ship	class_rel(Relation, Class1, Class2)	<owl:Class rdf:about="#Class1"> <owl:Relation rdf:re- source="#Class2"/> </owl:Class>
Instance- to- Instance Relation- ship	instance_rel(Relation, Instance1, Instance2)	<owl:ObjectProperty rdf:ID="Rel"> <rdfs:domain rdf:resource="#Obj1"/> <rdfs:range rdf:resource="#Obj2"/> </owl:ObjectProperty>
Instance- to-Class Relation- ship	instance_att(Instance, AttributeName, AttributeValue)	<owl:DatatypeProperty rdf:ID="Rel"/> <rdfs:domain rdf:resource="#Obj1"/> <rdfs:range rdf:re- source="#xsd:string"/> </owl:DatatypeProperty>

Figure 2.20: Mapping classes, instances and relationships between FBPML and OWL-S [53]

Finally, Figure 2.21 summarizes the mapping between FBPML and OWL-S.

Primitive	FBPML	OWL-S
Main Nodes	Activity	Composite Process
	Primitive Activity	Atomic Process
	Role	Participant
	Time Point	See <i>Note</i>
Links	Precedence Link	(part of) Sequence
	Synchronisation Bar	See <i>Note</i>
Junctions	Start	See <i>Note</i>
	Finish	See <i>Note</i>
	And-Joint	Split-Join
	Or-Joint	See <i>Note</i>
	And-Split	Split
	Or-Split	Repeat-While, Repeat-Until
	Xor-Junction	Choice
Annotations	Idea Note	See <i>Note</i>
	Navigation Note	See <i>Note</i>
Process Components	Precondition	Precondition
	Trigger	See <i>Note</i>
	Postcondition	Effect
	Precondition, Trigger and Postcondition	Input/Output
	Action	Atomic Process
	Conditional Action	If-Then-Else
<i>Note:</i> Limited (or no) equivalent convention provided by OWL-S.		

Figure 2.21: Summary of mapping FBPML and OWL-S process primitives [53]

As far as the limitations of the mapping are concerned, the process model components can only be partially translated [53]. Therefore, the translation between FBML and OWL-S is partial, because there are some elements that exist in FBPML but not in OWL-S and incomplete, because some of the translation cannot be conducted due to lack of knowledge about an element that is still in progress [53]. Thus, formal mapping between FBPML and OWL-S will require more work and research before a real-working mapping can be formulated.

2.5 Enabling Disciplines

At this point, we review two research areas that have some enabling characteristics that can help us address our research objectives. Specifically, the enabling disciplines are:

1. **Systems Theory**, providing critical concepts, and principles that help us integrate high-level process descriptions and low-level workflow execution semantics, maintaining consistency between the two different levels. Systems Theory helps us visualize and model process descriptions and execution semantics in one united form, as a set of elements standing in interrelation among themselves and with business environment [159].
2. **Constraint-based reasoning** provides us the means to model nonlinear relationships between business processes. Constraint-based reasoning is ideal for expressing and reasoning about business dynamics, while approaches like traditional programming constructs (if-then-else, while etc.) create extra complexity and make it difficult to easily change and adapt a process on a business demand.

2.5.1 Systems Theory

Systems theory was proposed in the 1940's by the biologist Ludwig von Bertalanffy [159] and furthered by Ross Ashby [160] and others. Ludwig von Bertalanffy, reacting against reductionism, underlined that real systems not only interact with their environments, but they can also obtain new properties through emergence, resulting in continual evolution [159]. Systems theory focuses on the arrangement of and relations between the parts or elements of an entity and connects them into a whole. *"This organization determines a system, which is independent of the concrete substance of the elements. Thus, the same concepts and principles of organization underlie the different disciplines (physics, biology, technology, sociology, etc.), providing a basis for their unification"* [161]. The concepts of a system include system-environment boundary, input, output, process, state, hierarchy, goal-directedness, and information [159].

According to the system concept, as defined in General System Theory [159], *"a system may be defined as a set of elements standing in interrelation among themselves and with environment. Progress is possible only by passing from a state of undifferentiated wholeness to a differentiation of parts"*. There are two categories of systems; open and closed. A system is 'closed' if no material enters or leaves it. On the other hand, a system is 'open' if there is import and export of material. More formally, *"an open system is defined as a system in exchange of matter with its environment, presenting import and export, building-up and breaking-down of its material components"* [159].

Among the aims of the General System Theory discussed by Bertalanffy [159], we focus on the tendency towards integration and the development of unifying principles running 'vertically' through the universe of the individual sciences, bringing us nearer to the goal of the unity of science. This aim matches our research goal to integrate high-level process descriptions and low-level execution semantics. Therefore, accommodating Systems Theory paradigm within our research context, we deal with open systems as we have exchange of business goals, needs, and demands, thus import and export of information. Through this interaction with business environments, workflow processes can acquire new characteristics, resulting in continual evolution. Moreover, rather than reducing a high-level process description to its execution parts, following systems theory principles we focus on the arrangement of and relations between the parts, which connect high-level process descriptions and low-level execution semantics into a whole. Finally, systems concepts of system-environment boundary, input, output, process, state, and hierarchy will help us towards the integration we are seeking for in our research.

2.5.2 Constraint-based reasoning

Constraint-based reasoning is an approach for problem solving that is based on deductive reasoning. In order to solve a problem using constraint-based reasoning, the problem is first modelled in terms of hypotheses and conclusion constraints. Then the problem is solved via constraint satisfaction [162]. Constraint-based reasoning has connections to a wide variety of fields and applications including design problems, scheduling and planning, causal reasoning, language understanding, qualitative and diagnostic reasoning, expert systems etc. [163].

Common practice in building and maintaining a business process is the construction of linear sequences of its activities from an inherently two, three, or multi-dimensional world [164]. Building models using linear approaches often creates misalignment with the business needs. The business world is nonlinear, and since processes are a way of describing the business world, process descriptions need to be nonlinear [164]. Some researchers have incorporated constraints with business process modelling. Specifically, Crampton [165] identifies a generic class of constraints, called entailment constraints, which restrict the execution order of process tasks with respect to authorization. Therefore, we consider constrained-based reasoning as an enabling paradigm that can help us effectively express and reason about business rules that capture nonlinear relationships.

Moreover, apart from effective process modelling of business, constraint-based reasoning can also help in maintenance of process models. Current modelling approaches explicitly sequence activities of workflow processes using traditional programming constructs (if-then-else, cases, while-loops etc.). When the process needs to be adapted, modellers have to resolve all the sequence and dependencies across its activities. Then they need to add, delete or update existing activities and at the end reallocate dependencies and sequence across them. On the other hand, encoding business rules by using constraints reduces the interdependencies across the activities of the workflow process and enables modellers to focus only on the activities that needs to be adapted. Thus, constrained-based reasoning can help us address changing conditions within workflow processes.

2.6 Conclusions

Below we give a schematic overview of workflow process modelling and execution research areas from the perspective of the literature review:

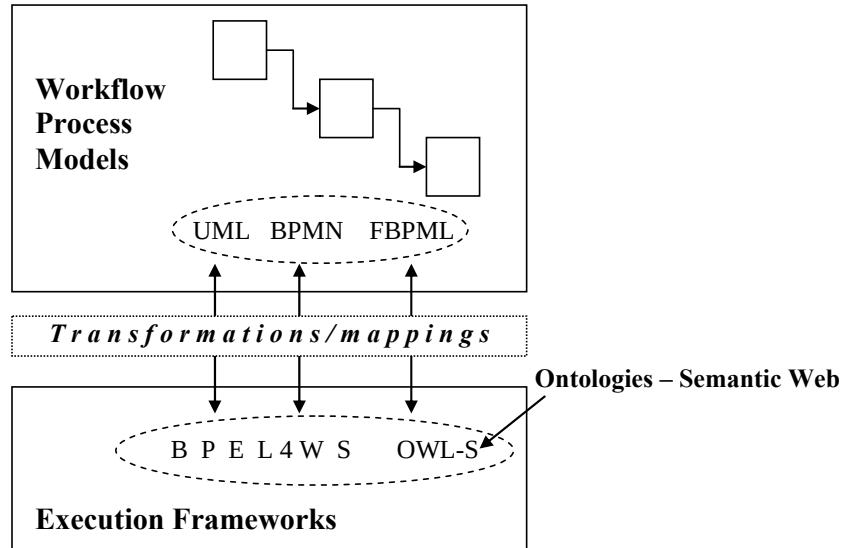


Figure 2.22: Rich picture of the Workflow Technology area

Specifically, as shown in Figure 2.22 there are two basic layers, namely modelling and execution layer. With regard to modelling layer, there are workflow modelling languages that capture business processes and create high-level visual process models. The workflow modelling languages reviewed in this Chapter were IDEF0, EPC, IDEF3, BPMN, XPD, BPD, and UML 2 ADs. Regarding the execution layer, we have reviewed BPEL4WS, OWL-S, and WSMX. Execution frameworks are able to define the low-level execution semantics for the processes that will be eventually executed by an execution engine. At this point, semantic web community with OWL-S and WSMX promote the use of ontologies, elaborating processes with semantics, in order to enable automatic discovery, composition, and execution of processes with little or even no human intervention.

Recent research approaches have defined mappings between the two layers trying to reduce the gap between them. The mappings we have reviewed were UML to BPEL4WS, BPMN to BPEL4WS and FBPML to OWL-S. These mappings try to address the existence of two different representations for high-level models and execution semantics, finding common constructs and concepts in order to provide an automatic mechanism of translating modelling languages to execution ones and vice versa. Although these mappings are positive approaches towards the vision of an integrated framework, they fail to fulfil a number of key requirements, as identified in [52]:

- Completeness, i.e. the mapping is only applicable to UML/BPMN/FBPML models that fulfil certain requirements.
- Automation, i.e. they are not capable of producing target code without requiring human intervention to identify patterns in the source model.
- Readability, i.e. the produced target code is not always understandable by humans; e.g., the produced BPEL4WS code due to automatic translation via the mappings becomes too complicated and often rather unreadable, obstructing refinement [52].

The main limitations we have identified in the literature in respect with our research objectives are as follows:

1. IDEF0 and IDEF3 are the only modelling languages that have formal semantics defined in its specification. The rest of modelling languages are either not formalized at all or have some proprietary formalization efforts trying to define some formal semantics but still are not finalized and not widely accepted. Absence of formalization results to ambiguous models that cannot be automatically checked for consistency and completeness.
2. There is no modelling language efficiently handling process change, in terms of quick adaptation of an existing model. Specifically, all process modelling languages we have reviewed explicitly sequence activity flow in a way that the final model includes a set of hard-connected activities. In case a modeller needs to add, change, or remove an activity, he has to find all the dependent activities and update them accordingly. It is obvious that in small and medium-sized models this would work but large scale and complex models would be very difficult to maintain.
3. Process modelling languages with lot of constructs may be more complete and accurate but at the same time require significant amount of training and access to technical resources in order to create models without errors.
4. Process modelling languages make no assumptions about the implementation of a process, thus they miss critical information for the execution logic needed by execution frameworks.
5. Process execution frameworks do not define graphical representation of processes nor provide any particular design methodology.
6. Process execution frameworks rely on either XML language or ontologies for the description of processes and their execution logic. Those that use XML cannot describe complex relationships and inheritance among activities in a process, while those using ontologies are difficult to learn and result to large descriptions that are difficult to read and write.
7. Business rules are mainly encoded at the execution layer using traditional programming constructs such as if-then-else, cases, while, etc. and this makes maintenance of process models and execution semantics even more difficult when process needs to be changed.

8. Although there are some proposals for mappings, common practice is still managing high-level modelling and low-level execution descriptions manually, creating difficulties in maintaining consistency between the two.
9. There is a lack an integrated modelling approach that could effectively represent high-level models and low-level execution semantics, supporting both layers of modelling in a smooth and consistent way.

Table 2.3 recapitulates the main limitations.

Modelling Languages
No Formal semantics (<i>EPC, BPMN, XPDL, BPDM, UML2 ADs</i>)
Sequencing activity flow (<i>EPC, IDEF3, BPMN, XPDL, BPDM, UML2 ADs</i>)
Lots of modelling constructs - significant amount of training and access to technical resources needed (<i>BPMN</i>)
No execution semantics (<i>IDEF0, EPC, IDEF3, BPDM, UML2 ADs</i>)
Execution Frameworks
No graphical representation
Sequence business rules with traditional programming constructs
Mappings (UML to BPEL4WS, BPMN to BPEL4WS, FBPML to OWL-S)
Completeness / Automation / Readability

Table 2.3: Main limitations identified in the literature

Finally, we reviewed two enabling disciplines for our research, namely Systems Theory and Constraint-based reasoning. Systems Theory has some enabling characteristics that can help us integrate high-level process descriptions and low-level workflow execution semantics into a unified format. On the other hand, constraint-based reasoning has some enabling characteristics that can help us effectively express and reason about business rules that capture nonlinear relationships in workflow processes and address changing conditions.

3 SYMEX: A Systems Theory based Framework for Workflow Modelling and Execution

3.1 Introduction

Both process modelling languages and workflow execution frameworks aim to support and automate business processes in real operational environments, at different levels of detail. Workflow processes in business environments usually incorporate a great deal of complexity due to the nature of business. They include both the essential physical activities and the essential management decision-making activities that control the flow logic of processes. Thus, based on the literature review, this Chapter defines a set of requirements that an integrated approach should satisfy in order to be able to effectively capture and represent high-level descriptions and low-level execution semantics of workflow processes.

In Chapter 2, we argued that the principles of Systems Theory could form a basis for rationalising the complexity of workflow processes and help us to derive appropriate constructs for dealing with that complexity. Here we describe the role of Systems Theory for defining an integrated approach for process modelling by identifying a set of interrelationships that exist between high-level process descriptions and low-level workflow execution semantics. By capturing this complex set of interrelationships we are able to support a reasoning about tracking changes in high-level descriptions due to internal and external conditions, and tracing the impact of those changes on the low-level execution semantics.

As a result, we address research objective RO1 by proposing a Systems Theory based framework, which integrates high-level process descriptions and low-level execution semantics, called **SYMEX** (**SY**stems theory based framework for workflow **M**odelling and **EX**ecution). More specifically, we define the basic concepts of the proposed framework, the rules that apply, and the corresponding formal representations. Finally, we analyse the frameworks modelling capability of capturing high-level descriptions and low-level execution semantics by examining the level of compliance with the set of requirements defined at the beginning of the Chapter.

The structure of this Chapter is as follows. Section 3.2 reviews the requirements for an integrated approach. Section 3.3 discusses the role of Systems Theory for defining such approach and defines the concepts, rules, and mathematical descriptions of SYMEX framework. Then, Section 3.4 examines the capability of SYMEX to support high-level process descriptions and low-level workflow execution semantics and Section 3.5 makes an overall comparison between SYMEX and other workflow modelling approaches. Finally, Section 3.6 concludes the Chapter with the overall findings.

3.2 Requirements for an Integrated Approach

An integrated modelling and execution approach should effectively satisfy the requirements for capturing and representing high-level descriptions and low-level execution semantics of workflow processes.

According to the literature [166], [51], [99], [167], [10], [165], [10], [168], [169], [170], [171], [172] and our previous work [173] five aspects of high-level descriptions of workflow processes can be regarded as representative set of minimum requirements that should be supported by an integrated approach:

1. **Flow representation [166], [51]:** to capture transfer of knowledge, information, goods from one point of the process to another. Flow also models the sequence of a process and is conducive to the way business analysts model processes [174].
2. **Conditions and effects [99], [167], [10]:** to capture preferences, preconditions, rules and results of business activities.
3. **Role assignment [165], [10], [168]:** to capture the different responsibilities of business roles in different parts of a business process model.
4. **Hierarchy representation [169], [170], [171]:** to capture separation of concerns at the business level, revealing the relations between the different parts of a process and providing a natural way for business people to categorize and make hierarchy of their business processes.
5. **Feedback loops [172]:** to capture the ability of a workflow process to react in response to demands by business environments. As business practices are developed or enhanced, the improvements can be incorporated into the workflow model as well, providing a feedback loop between the operations staff and the model. In the end, the model becomes a live, dynamic blueprint that captures knowledge about a complete distributed system in terms of its structure, behaviour, and characteristics [175].

The high-level business semantics presented above, should be integrated with low-level descriptions of workflow processes in order to be able to encode typical control flow dependencies encountered in workflow execution. Such dependencies can easily be identified analysing the control flow constructs existing in workflow execution languages. There is, indeed, a research initiative in workflow management presenting a set of workflow patterns [176], which provide a way to examine various perspectives of workflow execution semantics (e.g. control flow, data, resource, and exception handling). These patterns can be grouped in a number of categories as listed in the table below.

No	WF pattern description
Basic Control Patterns	
WP1	Sequence - execute two or more activities in sequence
WP2	Parallel Split - execute two or more activities in any order or in parallel
WP3	Synchronization - synchronize two or more activities that may execute in any order or in parallel
WP4	Exclusive Choice - choose one execution path from many alternatives based on data that is available when the execution of the process reaches the exclusive choice
WP5	Simple Merge - wait for one among a set of activities completes before proceeding
Advanced Branching and Synchronization Patterns	
WP6	Multiple Choice - choose several execution paths from many alternatives
WP7	Synchronizing Merge - merge many execution paths; synchronize if many paths are taken
WP8	Multiple Merge - wait for one among a set of activities to complete before proceeding
WP9	Discriminator - wait for one among a set of activities to complete before proceeding
Structural Patterns	
WP10	Arbitrary Cycles - do not impose any structural restrictions on the types of loops that can exist in the process model.
WP11	Implicit Termination - terminate an instance of the process if there is nothing else to be done
Multiple Instances (MI)	
WP12	MI without Synchronization - generate many instances of one activity without synchronizing them afterwards
WP13	MI with a Priori Design Time Knowledge - generate many instances of one activity when the number of instances is known at the design time
WP14	MI with a Priori Runtime Knowledge - generate many instances of one activity when a number of instances can be determined at some point during the runtime
WP15	MI without a Priori Runtime Knowledge - generate many instances of one activity when a number of instances cannot be determined
State-based patterns	
WP16	Deferred Choice - execute one of a number of alternatives threads.
WP17	Interleaved Parallel Routing - execute a number of activities in any order, but do not execute any of these activities at the same time/simultaneously.
WP18	Milestone - allow a certain activity at any time before the milestone is reached
Cancellation Patterns	
WP19	Cancel Activity - stop the execution of an enabled activity
WP20	Cancel Case - stop the execution of a running process

Table 3.1: Workflow patterns [104]

An integrated approach should be able to capture and represent low-level execution semantics of workflow processes. For this purpose, we adopt workflow patterns as a representative set of requirements.

3.3 Theoretical Framework

3.3.1 The Role of Systems Theory

In Chapter 2, we briefly reviewed General Systems Theory and discussed its importance for devising an integrated workflow modelling and execution framework, focusing on how its principles can be applied to address the business dynamics at different levels of abstractions, i.e. high-level descriptions and low-level execution semantics. This Section describes the role of Systems Theory for defining an integrated approach for process modelling, by identifying a set of interrelationships that exist between high-level process descriptions and low-level workflow execution semantics. By capturing this complex set of interrelationships we will be able to support a reasoning about tracking changes in high-level descriptions due to internal and external conditions, and tracing the impact of those changes on the low-level execution semantics. Thus, the central goal of adopting a systems perspective is to understand the complexity which arises from the two different levels of abstraction, in order to be able to define a formal description of how they behave as a whole, emerged from the interactions of their parts, i.e. process models and workflow execution semantics.

Systems Theory defines a system as a set of elements standing in interrelation among them and with environment. The Theory focuses on the arrangement of and relations between the parts or elements of an entity and connects them into a whole. Moreover, Ludwig von Bertalanffy, suggested among other things the existence of system isomorphism [159], i.e. the existence of general mathematical descriptions explaining the dynamic behaviour of very different kinds of systems at different scales, thus introducing hierarchy for determining multiple layers with different detail of description. Inspired by these concepts, we consider a workflow model as an 'entity' and at the same time high-level and low-level executable descriptions as two 'elements' of this 'entity'. So the problem of integration becomes a problem of defining the arrangement of and the relations between these two 'elements' and their parts. The solution is to conceptualize the high-level description as the upper layer 'element' and the low-level execution semantics as the lower layer 'element' of a hierarchical decomposition of the 'entity' workflow model, as shown in Figure 3.1. Both elements are thus connected into a whole, inside the 'entity' workflow model.

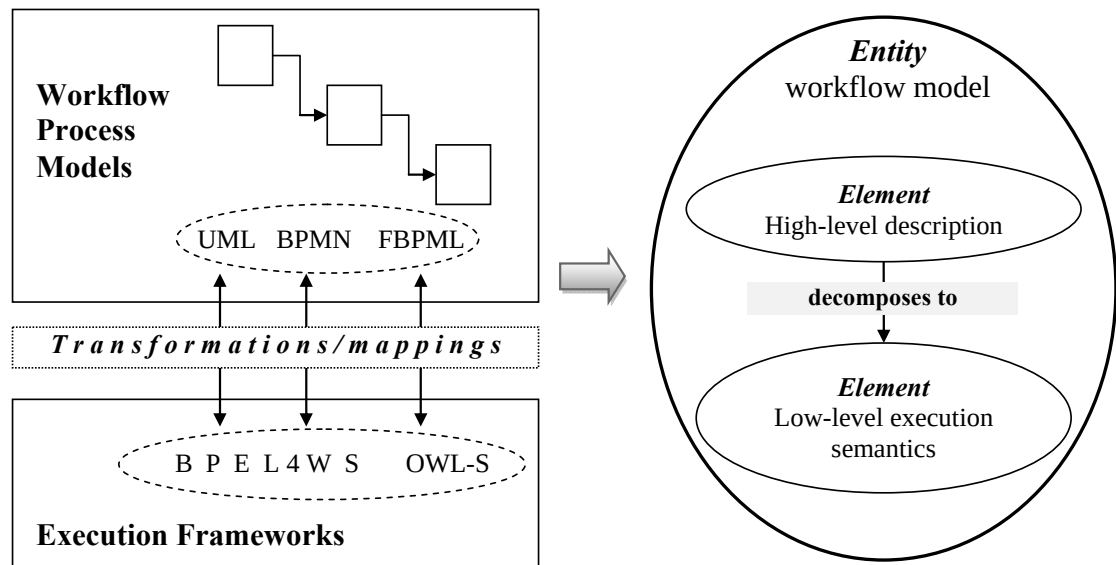


Figure 3.1: Transforming the problem of integration using Systems Theory

3.3.2 Definition of Concepts

A business process consists of a network of resources, actors, process steps and the interactions between them. Therefore, in order to model business processes under the workflow context, we adopt a systems perspective of the process-centred workflows and analyse them separately from the environment by defining their boundaries. **Workflow boundaries** are defined with respect to the **workflow goals** and the **business constraints**. Workflow goals express the expectations of various agents about the ability of the workflow to carry the desired process execution. On the other hand, business constraints affect workflow, posing requirements for the delivered outputs and the inputs that may be used to deliver this output. Both workflow goals and business constraints define the scope of the workflow model and execution.

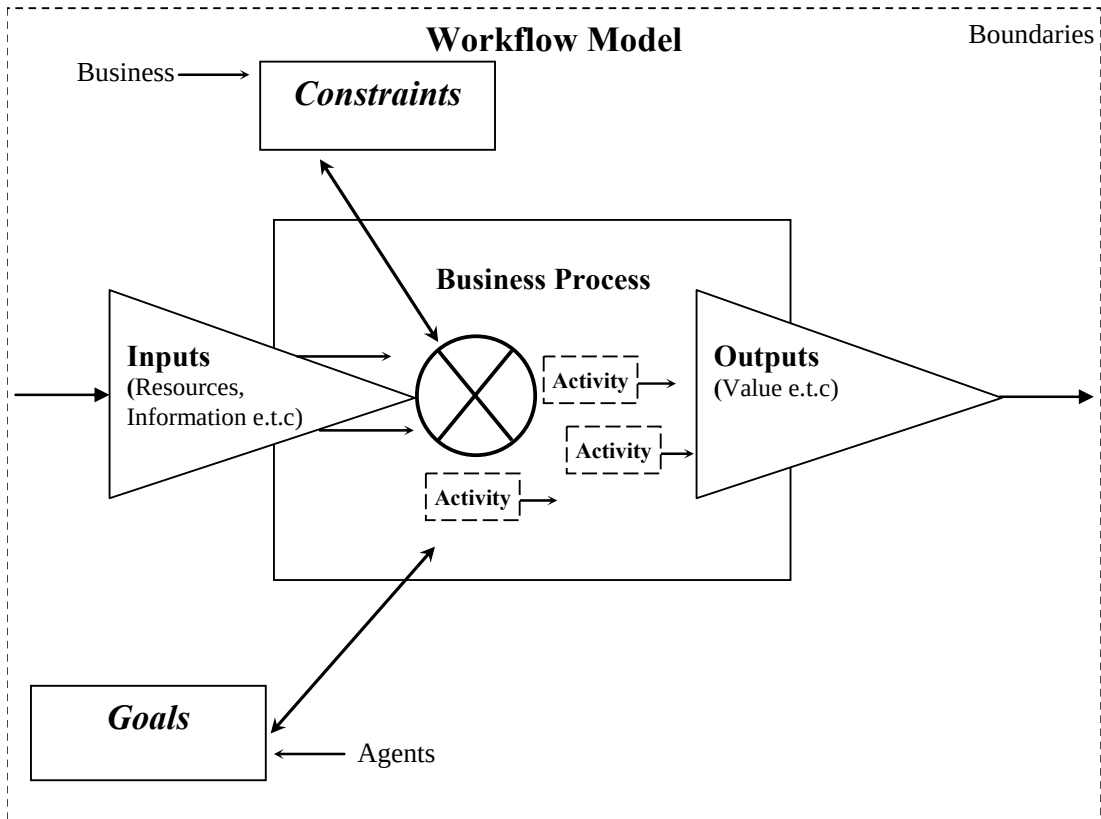


Figure 3.2: A Systemic perspective of a workflow model

Figure 3.2 represents the highest level of abstraction of a workflow model. Subsequently, realization of a workflow process is perceived as the coordination of a network of activities organized in the context of a hierarchy. Thus, modelling the workflow process from a Systems Theory perspective, we consider it as a system of interconnected *activities* with *inputs*, *outputs*, *controls*, and *mechanisms*. More specifically, finalizing the model we introduced in an initial form in [173], that is greatly influenced by IDEF0 modelling technique, we view a workflow process, as a set of **Activities**, which take **Inputs** and produce **Outputs**. Activities are executed via the means of their **execution Mechanisms** and activated by their **Control**, which evaluates a number of constraints. Figure 3.3 shows a visual notation we propose for communication purposes.

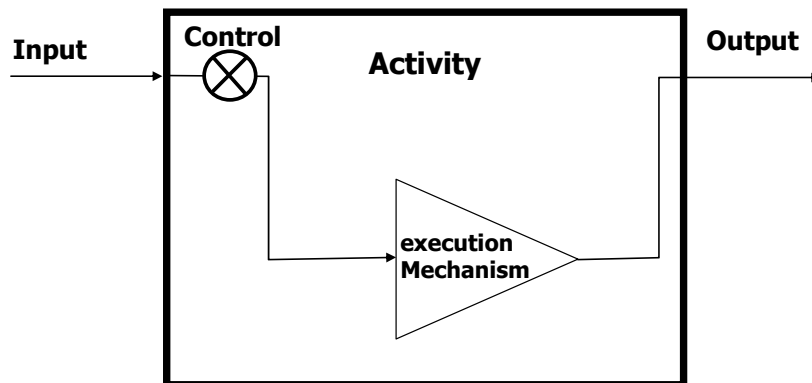


Figure 3.3: Visual notation of SYMEX models

We argue that these constructs provide sound foundations for expressing business dynamics in high-level descriptions as well as the semantics of low-level process execution, and we will validate this argument in Section 3.4.

At this point we logically define the concepts of our proposed framework, based on the concepts introduced in [173]:

1. **Activity**: The main construct of our formalism is the Activity. A workflow process model is composed by a number of activities. According to the level of decomposition, the activity can represent the whole process, a sub-process, or an atomic activity. There are two types of Activities; ***composite***, that are decomposed further to sub-activities and ***atomic***, that are not decomposed.
2. **Inputs**: Inputs of Activities are structures of information elements related to software components involved in the work activities of the workflow process.
3. **Outputs**: Outputs of Activities are structures of information elements related to software components involved in the work activities of the workflow process.
4. **Controls**: Controls are logical conditions and constraints that control the execution of the Activity.
5. **Execution Mechanisms**: Execution Mechanisms are the means by which Activities are being executed using Inputs and producing the Output. Execution Mechanisms are software components, web-services, or business roles responsible for the realization of the Activity.

3.3.3 Mathematical Descriptions

Having provided an informal definition of the main concepts of our framework, we continue in this Section with their mathematical descriptions. These descriptions are mainly based on a set theory that allows to reason about the process characteristics at different levels of detail, in a more controlled way.

3.3.3.1 Basic constructs

Derived from our formalization efforts in [177], an **Activity A** is defined as a **4-tuple (I, C, O, M)** where:

- **I** is a set of structures of information elements $I_1, I_2, \dots, I_n$ [defined as **$I(A)$**], where n is an integer number.

$$I(A): \forall i, 1 \leq i \leq n, I_i \in I(A)$$

Formula 3.1: Definition of $I(A)$ – Inputs of Activity

- **O** is a set of structures of information elements $O_1, O_2, \dots, O_n$ [defined as **$O(A)$**], where n is an integer number.

$$O(A): \forall i, 1 \leq i \leq n, O_i \in O(A)$$

Formula 3.2: Definition of $O(A)$ – Outputs of Activity

- **C** is a set of *control rules* $C_1, C_2, \dots, C_n$ [defined as **$C(A)$**], where n is an integer number.

$$C(A): \forall i, 1 \leq i \leq n, C_i \in C(A)$$

Formula 3.3: Definition of $C(A)$ – Controls of Activity

- **M** is a set of execution mechanisms $M_1, M_2, \dots, M_n$ [defined as **$M(A)$**], where n is an integer number.

$$M(A): \forall i, 1 \leq i \leq n, M_i \in M(A)$$

Formula 3.4: Definition of $M(A)$ – execution Mechanisms of Activity

Based on the above definitions, a workflow process is set **Σ** consisting of activities **$A_1, A_2, \dots, A_n \in \Sigma$** having inputs **$I(A)$** , outputs **$O(A)$** , controls **$C(A)$** and execution mechanisms **$M(A)$** .

An Activity can be either atomic or composite. Therefore, a workflow process **Σ** is defined as **$\Sigma = \text{AtAct}(\Sigma) \vee \text{CAct}(\Sigma)$** , where **AtAct** is the set of **atomic Activities** and **CAct** is the set of **composite Activities**.

3.3.3.2 Modelling semantics

As far as the modelling is concerned, we define the following rules that apply to SYMEX models:

1. An Activity A_i must have at least one Input and up to n Inputs, where n is an integer number. Based on Formula 3.1 we have:

$$\exists I_k \in I(A_i), 1 \leq k \leq n, 1 \leq i \leq n, A_i \in \Sigma$$

Formula 3.5: Rule 1

2. An atomic Activity A_i must have exactly one Output. Based on Formula 3.2 we have:

$$\exists O_k \in O(A_i), k = 1, 1 \leq i \leq n, A_i \in \Sigma, A_i \in \text{AtAct}(\Sigma)$$

Formula 3.6: Rule 2

3. A composite Activity A_i must have at least one Output and up to n Outputs, where n is an integer number. Based on Formula 3.2 we have:

$$\exists O_k \in O(A_i), 1 \leq k \leq n, 1 \leq i \leq n, A_i \in \Sigma, A_i \in \text{CAct}(\Sigma)$$

Formula 3.7: Rule 3

4. An atomic Activity A_i may have zero to n Controls, where n is an integer number. Based on Formula 3.3 we have:

$$(C(A_i) = \emptyset \vee (\exists C_k \in C(A_i), 1 \leq k \leq n)), A_i \in \Sigma, A_i \in \text{AtAct}(\Sigma), 1 \leq i \leq n$$

Formula 3.8: Rule 4

5. An atomic Activity A_i may have zero to n Execution Mechanisms, where n is an integer number. Based on Formula 3.4 we have:

$$(M(A_i) = \emptyset \vee (\exists M_k \in M(A_i), 1 \leq k \leq n)), A_i \in \Sigma, A_i \in \text{AtAct}(\Sigma), 1 \leq i \leq n$$

Formula 3.9: Rule 5

6. Outputs of Activities may be used as Inputs to other activities. Based on Formula 3.1 and Formula 3.2 we have:

$$\exists A_i \in \Sigma, A_k \in \Sigma : (\exists j \in I(A_k) \wedge j \in O(A_i)), 1 \leq i \leq n, 1 \leq k \leq n$$

Formula 3.10: Rule 6

3.3.3.3 Execution Semantics

As far as the execution semantics are concerned, we give the following definitions:

1. An atomic Activity can be only in one of three possible states '**available**', '**unavailable**', or '**executed**'. **ActSt** is the set of states of activities $A_1, A_2, \dots, A_n \in \Sigma$, defined as **ActSt_{A1}, ActSt_{A2}, ActSt_{A3}, ..., ActSt_{An} [ActSt(Σ)]**, where Σ is the workflow process. **ActSt_{Ai}** is essentially a function having as input an Activity A_i and producing as output one of the values '**available**' or '**unavailable**' or '**executed**':

$$\begin{aligned} \text{ActSt}_{A_i} = f: A_i \rightarrow , A_i \in \text{AtAct}(\Sigma) \\ \text{and} \\ \text{ActSt}(\Sigma): \forall i, 1 \leq i \leq n, A_i \in \Sigma, \text{ActSt}_{A_i} \in \text{ActSt}(\Sigma), A_i \in \text{AtAct}(\Sigma) \end{aligned}$$

Formula 3.11: Definition of Activity States for a workflow process Σ

2. Each atomic Activity of a SYMEX model can be executed whenever it is '**available**'. An atomic Activity A_i is '**available**' when the Controls **C(A_i)** of atomic Activity A_i evaluate to true:

$$\begin{aligned} (\forall C_i \in C(A_i), \text{eval}(C_i) = \text{true}), 1 \leq i \leq n, A_i \in \text{AtAct}(\Sigma) \\ \Rightarrow \text{ActSt}_{A_i} = f: A_i \rightarrow \{\text{available}\} \end{aligned}$$

Formula 3.12: Definition of status 'available**' for Activity A_i**

3. Inputs are used by the execution Mechanisms of atomic Activities in order to produce their Output. Execution Mechanism **M(A_i)** is essentially a function having as input the Inputs **I(A_i)** of atomic Activity A_i and producing as output the Output **O(A_i)** of atomic Activity A_i :

$$\begin{aligned} \forall A_i \in \Sigma, A_i \in \text{AtAct}(\Sigma) : (\exists i \in I(A_i) \wedge o \in O(A_i) \wedge m \in M(A_i)) \\ \Rightarrow m = f: i \rightarrow o \end{aligned}$$

Formula 3.13: Definition of execution Mechanism of Activity A_i

4. The Inputs and the set of Activity States of a workflow process Σ are used by the Controls of atomic Activities to constrain their execution. **Control C(A_i)** of atomic Activity A_i is essentially a function having as input the Inputs **I(A_i)** of atomic Activity A_i and the set of Activity States of **ActSt(Σ)** of workflow process Σ and producing as output one of the values '**true**' or '**false**':

$$\begin{aligned} \forall A_i \in \Sigma, A_i \in \text{AtAct}(\Sigma) : (\exists i \in I(A_i) \wedge c \in C(A_i)) \\ \Rightarrow c = f: (i \wedge \text{ActSt}(\Sigma)) \rightarrow \{\text{True} \vee \text{False}\} \end{aligned}$$

Formula 3.14: Definition of Control of Activity A_i

3.4 Framework Analysis

SYMEX aims to bridge the gap between high-level process descriptions and low-level execution semantics. In the following Sections, we examine the capability of the proposed framework to capture the business semantics, which are identified in high-level workflow process descriptions. Then we examine the framework's capability to support low-level workflow execution semantics in terms of capturing a number of workflow patterns [104] that encode typical control flow dependencies encountered in workflow execution. The analysis is based on the framework's compliance with the set of requirements defined at Section 3.2 of this Chapter.

3.4.1 Business Semantics

In a high-level description of a workflow process, we have identified five distinct aspects that express the business semantics: (1) Flow representation, (2) Conditions and effects, (3) Role assignment, (4) Hierarchy representation and (5) Feedback loops. The first three aspects are easily modelled in SYMEX, as shown in Figure 3.4. Flow representation can be expressed with the concepts of Input and Output. Condition and effects can be expressed through concepts of Controls and Execution Mechanisms, while Role assignment through Execution Mechanism.

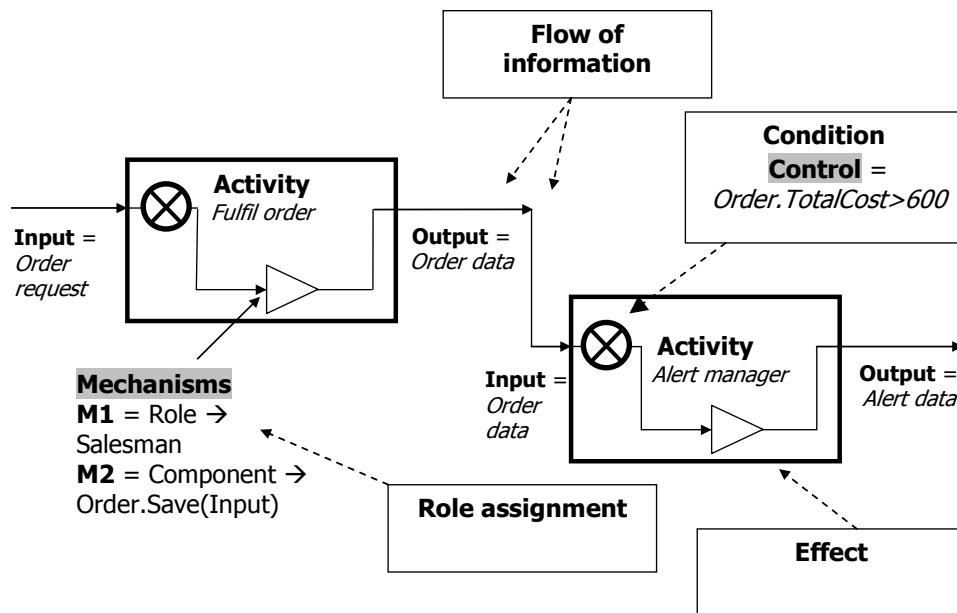


Figure 3.4: Modelling basic aspects of business semantics

The fourth aspect, which is Hierarchy representation, is only supported by a few workflow process modelling languages, i.e. IDEF0, IDEF3 and BPMN, and not supported at all by execution frameworks. Finally, the fifth aspect, which is Feedback loops, is implicitly supported by some process modelling languages and execution frameworks. In the next Sections, we examine whether SYMEX can support hierarchy representation and feedback loops respectively.

3.4.1.1 Hierarchy Representation

Hierarchy is a good modelling technique for separation of concerns. As separation of concerns starts at the business level and business people depending on their hierarchy and position want to have the right information at the right place and time, it would be easier for them to understand, communicate, and even participate in the design of models that would enable the separation of workflow processes in a hierarchical manner. Therefore, hierarchy representation in a model reveals the relations between the different parts of a process and provides a natural way for business people to categorize and make hierarchy of workflow processes that reflect their business. SYMEX supports hierarchy through the decomposition of composite Activities. Composite Activities inherit their Inputs, Outputs, Controls, and execution Mechanisms to the decomposed Activities.

Semantics: A composite Activity A_j has i as Input, o as Output, c as Control and m as execution Mechanism. The Activity A_j is decomposed to Activities $A_{j1}, \dots, A_{jn}$ if Input i , Output o , Control c and execution Mechanism m of Activity A_j are Input, Output, Control and execution Mechanism of at least of one of the Activities $A_{j1}, \dots, A_{jn}$, where n is an integer number:

$$\begin{aligned} \forall A_j \in \Sigma \exists A_{j1}, \dots, A_{jn} \in \Sigma : \\ & (\exists i \in I(A_j) \wedge o \in O(A_j) \wedge c \in C(A_j) \wedge m \in M(A_j)) \wedge \\ & (i \in I(A_{j1}) \vee \dots \vee i \in I(A_{jn})) \wedge \\ & (o \in O(A_{j1}) \vee \dots \vee o \in O(A_{jn})) \wedge \\ & (c \in C(A_{j1}) \vee \dots \vee c \in C(A_{jn})) \wedge \\ & (m \in M(A_{j1}) \vee \dots \vee m \in M(A_{jn})) \end{aligned}$$

Formula 3.15: Hierarchy's semantics

Example: An e-shop has a workflow process, which handles the order requests and ships the ordered items. In an abstract level, the Activity "Process order" is presented as having Input an "Order request" and as Output the "Items for shipping". This view hides the implementation details of Activity "Process orders" and only in the decomposed level we may observe the constituent Activities; "Check warehouse for items" and "Invoice order".

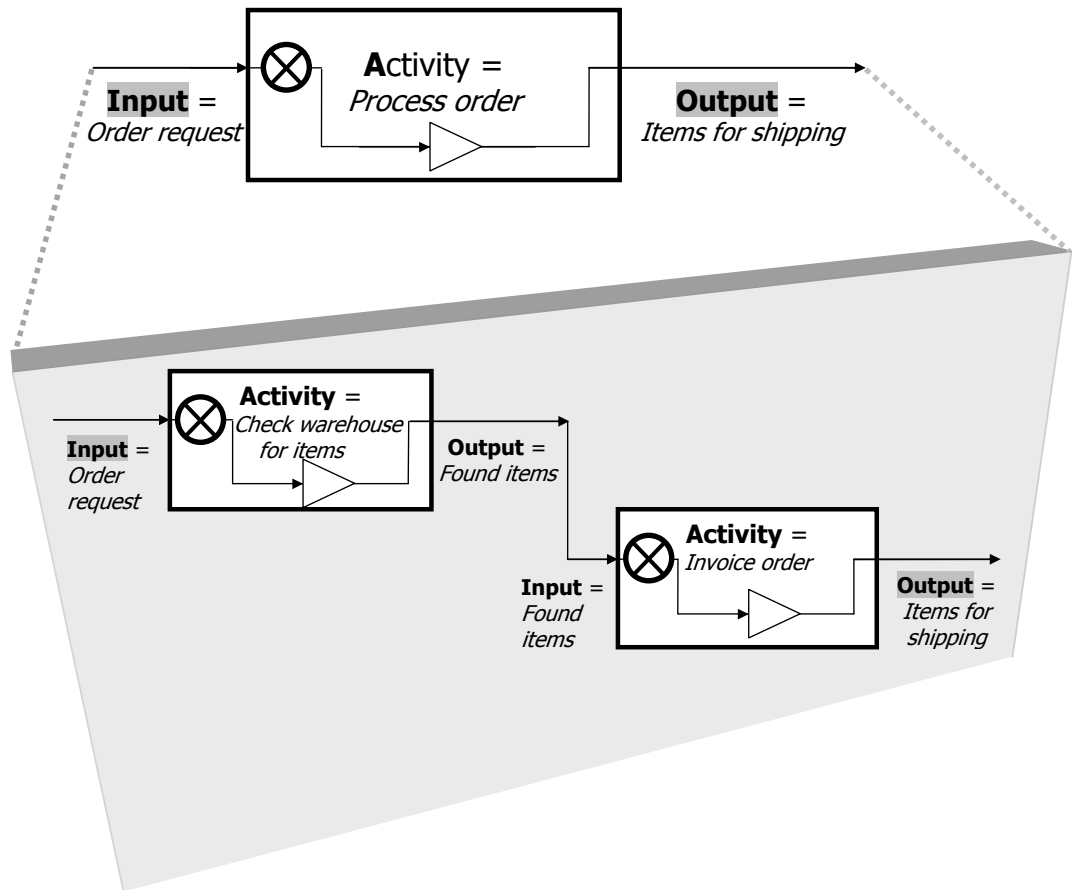


Figure 3.5: An example of hierarchy in a workflow process model

3.4.1.2 Feedback Loops

Feedback is a process whereby some proportion of the output of a system is passed (fed back) to the input of the system. As a workflow process changes due to demand by business environments, feedback loops control the execution and amplify possibilities of divergences with the new business goals [172].

Semantics: If an Activity A_j has i as Input, o as Output and c as Control, then Feedback loop $F(A_j)$ is defined as a subset of Output o , that feeds back the Control c of Activity A_j : Then, Control c of Activity A_j is essentially a function taking as input the UNION of Input i and Feedback loop $F(A_j)$ and producing as output value 'true' or 'false':

$$\begin{aligned} \exists A_j \in \Sigma : & \left(\exists i \in I(A_j) \wedge o \in O(A_j) \wedge c \in C(A_j) \right) \\ & \Rightarrow F(A_j) \subseteq o \in O(A_j) \\ & \Rightarrow c = f: (i \vee F(A_j)) \rightarrow \{\text{True} \vee \text{False}\} \end{aligned}$$

Formula 3.16: Feedback loop's semantics

Example: An investor holds a savings account. Every month he compares all available interest rates of the market and in case he finds a higher interest rate than his current one, he opens a new account and transfers all his money. The new interest rate feeds back the investor's monthly activity. Figure 3.6 shows the visual representation of this example.

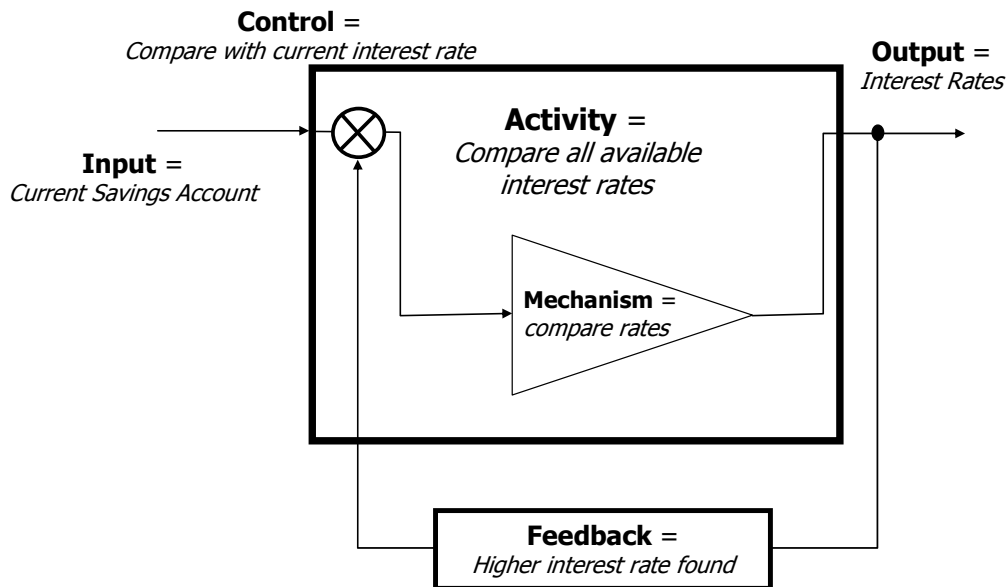


Figure 3.6: An example of feedback loop in a workflow process model

3.4.2 Workflow Process Execution Semantics

In the previous Section, we examined the capability of the proposed framework to formalize the business dynamics semantics that are identified in high-level workflow process descriptions. This Section examines the 20 workflow patterns presented in [104], and show how and to what extent these patterns can be captured using SYMEX modelling. The examples employed are the same with [104]. Most of the solutions are presented in a simplified model based on the concepts we introduced in Section 3.3, which is rich enough for capturing the key ideas of the solutions.

WP1 Sequence: An activity in a workflow process is executed after the completion of another activity in the same process.

Example: After the Activity “*order registration*” the Activity “*customer notification*” is executed.

Solution, WP1: The solution is depicted in Figure 3.7. Here we highlight on the Control concept, which constrains the execution of Activity A2, based on the completion of Activity A1.

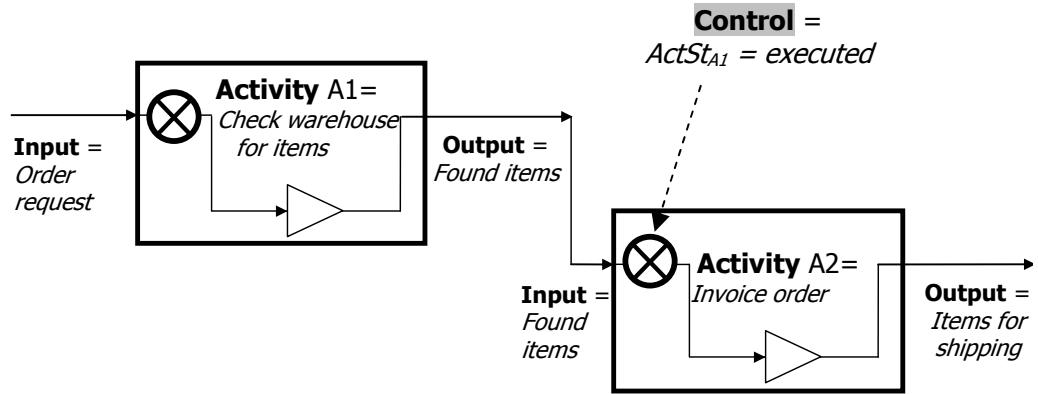


Figure 3.7: Workflow pattern 1: Sequence

$\exists A_1, A_2 \in \Sigma :$

$$\begin{aligned}
 & (I_{A1}(\text{Order request}) \in I(A_1) \wedge O_{A1}(\text{Found items}) \in O(A_1)) \wedge \\
 & (I_{A2}(\text{Found items}) \in I(A_2) \wedge O_{A2}(\text{Items for shipping}) \in O(A_2) \wedge C_{A2} \in C(A_2)) \wedge \\
 & (I_{A2}(\text{Found items}) = O_{A1}(\text{Found items})) \wedge \\
 & (C_{A2} = f : (ActSt_{A1} = \text{executed}) \rightarrow \{\text{True} \vee \text{False}\})
 \end{aligned}$$

Formula 3.17: Semantics of Workflow pattern 1: Sequence

WP2 Parallel Split: A point in the process where a single thread of control splits into multiple threads of control, which can be executed in parallel, thus allowing activities to be executed simultaneously or in any order.

Example: After Activity “*Subscribe new mobile phone*”, Activities “*Update Mobile Location Registry*” and “*Update Service Registry*” are executed in parallel.

Solution, WP2: The solution is depicted in Figure 3.8. The parallel split is realized by defining the Activities’ Controls as “ $ActSt_{A1} = \text{executed}$ ”, so that the activities are executed in parallel.

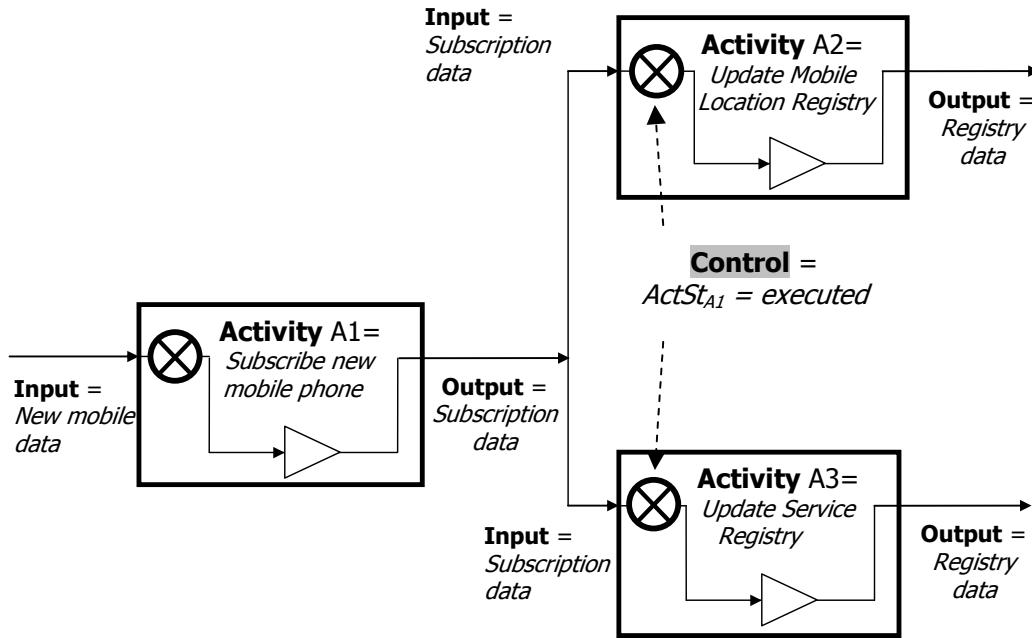


Figure 3.8: Workflow pattern 2: Parallel Split

$\exists A_1, A_2, A_3 \in \Sigma :$

$$\begin{aligned}
 & (I_{A1}(\text{New mobile data}) \in I(A_1) \wedge O_{A1}(\text{Subscription data}) \in O(A_1)) \wedge \\
 & (I_{A2}(\text{Subscription data}) \in I(A_2) \wedge O_{A2}(\text{Registry data}) \in O(A_2) \wedge C_{A2} \in C(A_2)) \wedge \\
 & (I_{A3}(\text{Subscription data}) \in I(A_3) \wedge O_{A3}(\text{Registry data}) \in O(A_3) \wedge C_{A3} \in C(A_3)) \wedge \\
 & (I_{A2}(\text{Subscription data}) = I_{A3}(\text{Subscription data}) = O_{A1}(\text{Subscription data})) \wedge \\
 & (C_{A2} = C_{A3} = f : (ActSt_{A1} = \text{executed}) \rightarrow \{\text{True} \vee \text{False}\})
 \end{aligned}$$

Formula 3.18: Semantics of Workflow pattern 2: Parallel Split

WP3 Synchronization: A point in the process where multiple parallel branches converge into one single thread of control, synchronizing multiple threads.

Example: Activity “Mail tickets” is executed after the completion of both Activities “Issue tickets” and “Issue Invoice”.

Solution, WP3: The solution is depicted in Figure 3.9. Activity “Mail tickets” has as Control “ActSt_{A1}=executed AND ActSt_{A2}=executed”.

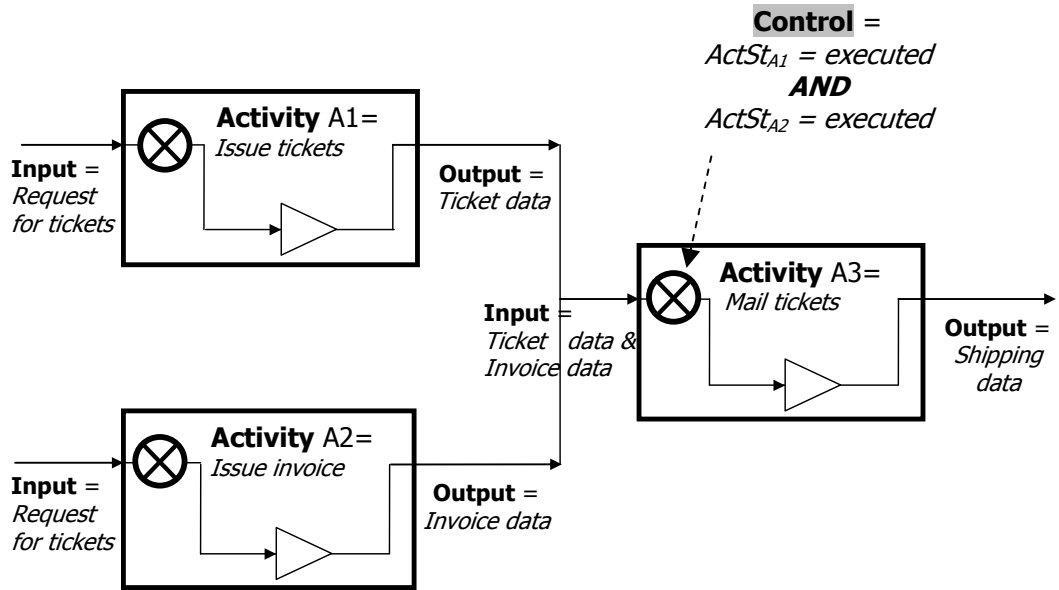


Figure 3.9: Workflow pattern 3: Synchronization

$\exists A_1, A_2, A_3 \in \Sigma :$

$$\begin{aligned}
 & (I_{A1}(\text{Request for tickets}) \in I(A_1) \wedge O_{A1}(\text{Ticket data}) \in O(A_1)) \wedge \\
 & (I_{A2}(\text{Request for tickets}) \in I(A_2) \wedge O_{A2}(\text{Invoice data}) \in O(A_2)) \wedge \\
 & (I_{A3}(\text{Ticket \& Invoice data}) \in I(A_3) \wedge O_{A3}(\text{Shipping data}) \in O(A_3) \wedge C_{A3} \in C(A_3)) \wedge \\
 & (I_{A3}(\text{Ticket \& Invoice data}) = O_{A1}(\text{Ticket data}) \wedge O_{A2}(\text{Invoice data})) \wedge \\
 & (C_{A3} = f : (\text{ActSt}_{A1}=\text{executed} \text{ AND } \text{ActSt}_{A2}=\text{executed}) \rightarrow \{\text{True} \vee \text{False}\})
 \end{aligned}$$

Formula 3.19: Semantics of Workflow pattern 3: Synchronization

WP4 Exclusive Choice: A point in the workflow process where, based on a decision or workflow control data, one of several branches is chosen.

Example: The manager is alerted if an order exceeds € 600, otherwise not.

Solution, WP4: The solution is depicted in Figure 3.10. Activity “*Alert manager*” has a Control that evaluates the cost of the order.

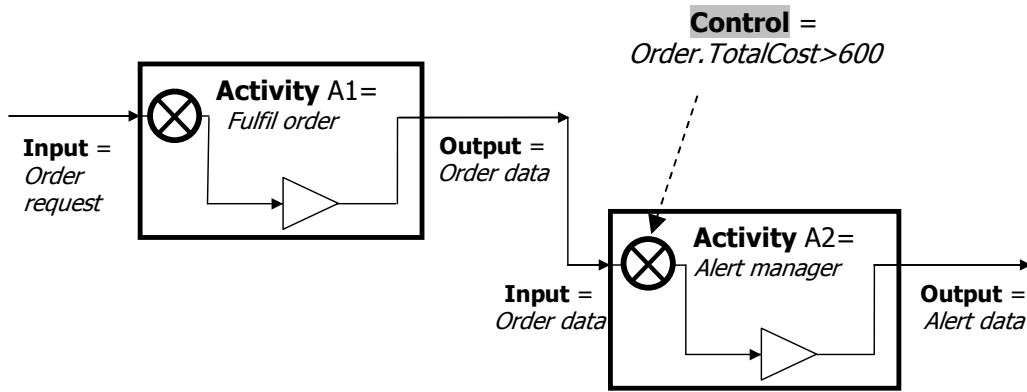


Figure 3.10: Workflow pattern 4: Exclusive Choice

$$\begin{aligned}
 &\exists A_1, A_2 \in \Sigma : \\
 &\quad (I_{A1}(\text{Order request}) \in I(A_1) \wedge O_{A1}(\text{Order data}) \in O(A_1)) \wedge \\
 &\quad (I_{A2}(\text{Order data}) \in I(A_2) \wedge O_{A2}(\text{Alert data}) \in O(A_2) \wedge C_{A2} \in C(A_2)) \wedge \\
 &\quad (I_{A2}(\text{Order data}) = O_{A1}(\text{Order data})) \wedge \\
 &\quad (C_{A2} = f: (\text{Order.TotalCost} > 600) \rightarrow \{\text{True} \vee \text{False}\})
 \end{aligned}$$

Formula 3.20: Semantics of Workflow pattern 4: Exclusive Choice

WP5 Simple Merge: A point in the workflow process where two or more alternative branches come together without synchronization.

Example: After Activity “*Receive payment*” is completed or Activity “*Approve bank loan*” is completed, then Activity “*Deliver car to customer*” is executed.

Solution, WP5: The solution is depicted in Figure 3.11.

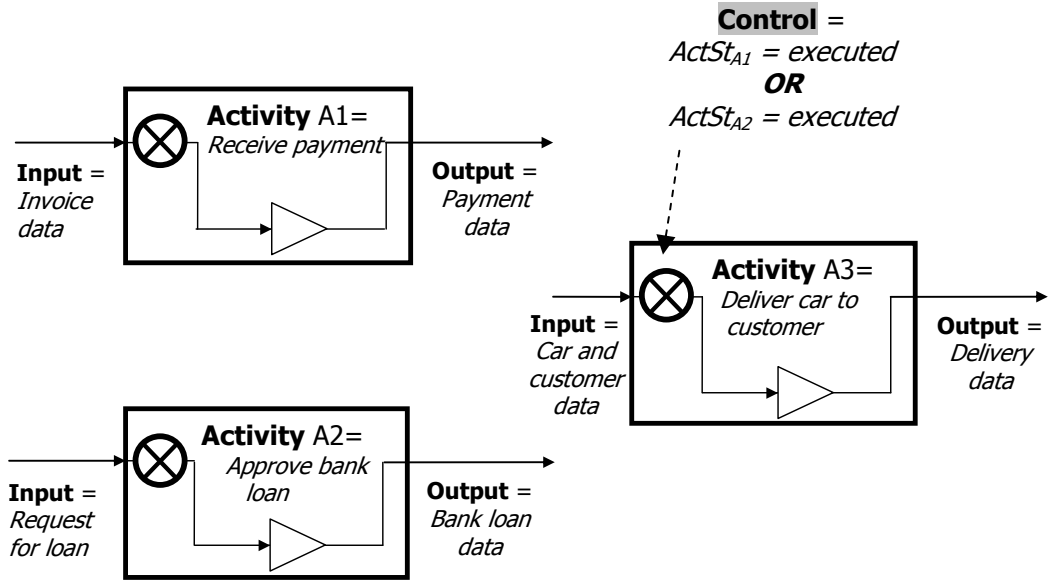


Figure 3.11: Workflow pattern 5: Simple Merge

$\exists A_1, A_2, A_3 \in \Sigma :$

$$\begin{aligned}
 & (I_{A1}(\text{Invoice data}) \in I(A_1) \wedge O_{A1}(\text{Payment data}) \in O(A_1)) \wedge \\
 & (I_{A2}(\text{Request for loan}) \in I(A_2) \wedge O_{A2}(\text{Bank loan data}) \in O(A_2)) \wedge \\
 & (I_{A3}(\text{Car \& customer data}) \in I(A_3) \wedge O_{A3}(\text{Delivery data}) \in O(A_3) \wedge C_{A3} \in C(A_3)) \wedge \\
 & (C_{A3} = f : (\text{ActSt}_{A1} = \text{executed} \text{ OR } \text{ActSt}_{A2} = \text{executed}) \rightarrow \{ \text{True} \vee \text{False} \})
 \end{aligned}$$

Formula 3.21: Semantics of Workflow pattern 5: Simple Merge

WP6 Multi-Choice: A point in the process, where, based on a decision or control data, a number of branches are chosen and executed as parallel threads.

Example: After completion of Activity “Evaluate damage”, Activity “Contact fire department” or Activity “Contact insurance company” is executed. At least one of these activities is executed. However, it is also possible that both need to be executed.

Solution, WP6: The solution is depicted in Figure 3.12. In case Activity “Evaluate damage” identifies a fire factor then the fire department is contacted. In addition, if the damaged property was insured, the insurance company is contacted. The Controls of Activities implement the multi-choice pattern.

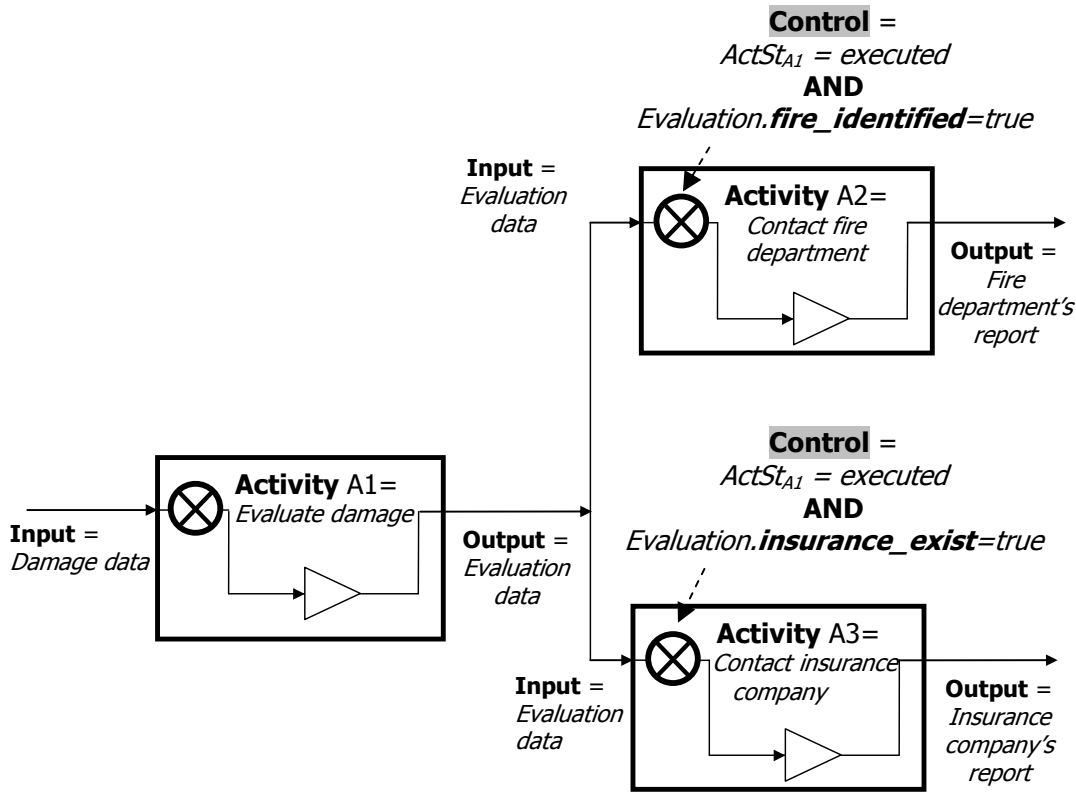


Figure 3.12: Workflow pattern 6: Multi-Choice

$$\begin{aligned}
 &\exists A_1, A_2, A_3 \in \Sigma : \\
 &\quad (I_{A1}(\text{Damage data}) \in I(A_1) \wedge O_{A1}(\text{Evaluation data}) \in O(A_1)) \wedge \\
 &\quad (I_{A2}(\text{Evaluation data}) \in I(A_2) \wedge O_{A2}(\text{Fire dept's report}) \in O(A_2) \wedge C_{A2} \in C(A_2)) \wedge \\
 &\quad (I_{A3}(\text{Evaluation data}) \in I(A_3) \wedge O_{A3}(\text{Insurance comp's report}) \in O(A_3) \wedge C_{A3} \in C(A_3)) \wedge \\
 &\quad (I_{A2}(\text{Evaluation data}) = I_{A3}(\text{Evaluation data}) = O_{A1}(\text{Evaluation data})) \wedge \\
 &\quad (C_{A2} = f : (\text{ActSt}_{A1} = \text{executed} \text{ AND } \text{Evaluation.fire_identified} = \text{true}) \rightarrow \{ \text{True} \vee \text{False} \}) \wedge \\
 &\quad (C_{A3} = f : (\text{ActSt}_{A1} = \text{executed} \text{ AND } \text{Evaluation.insurance_exist} = \text{true}) \rightarrow \{ \text{True} \vee \text{False} \})
 \end{aligned}$$

Formula 3.22: Semantics of Workflow pattern 6: Multi-Choice

WP7 Synchronizing Merge: A point in the process where multiple paths converge into one single thread. Some of these paths are "active" (i.e. they are being executed) and some are not. If only one path is active, the activity after the merge is triggered as soon as this path completes. If more than one path is active, synchronization of all active paths need to take place before the next activity is triggered. It is an assumption of this pattern that a branch that has already been activated, cannot be activated again while the merge is still waiting for other branches to complete.

Example: After either or both of the Activities "Contact fire department" and "Contact insurance company" have been completed, the Activity "Submit report" needs to be performed. Activity "Submit report" needs to be executed only once.

Solution, WP7 The solution is depicted in Figure 3.13.

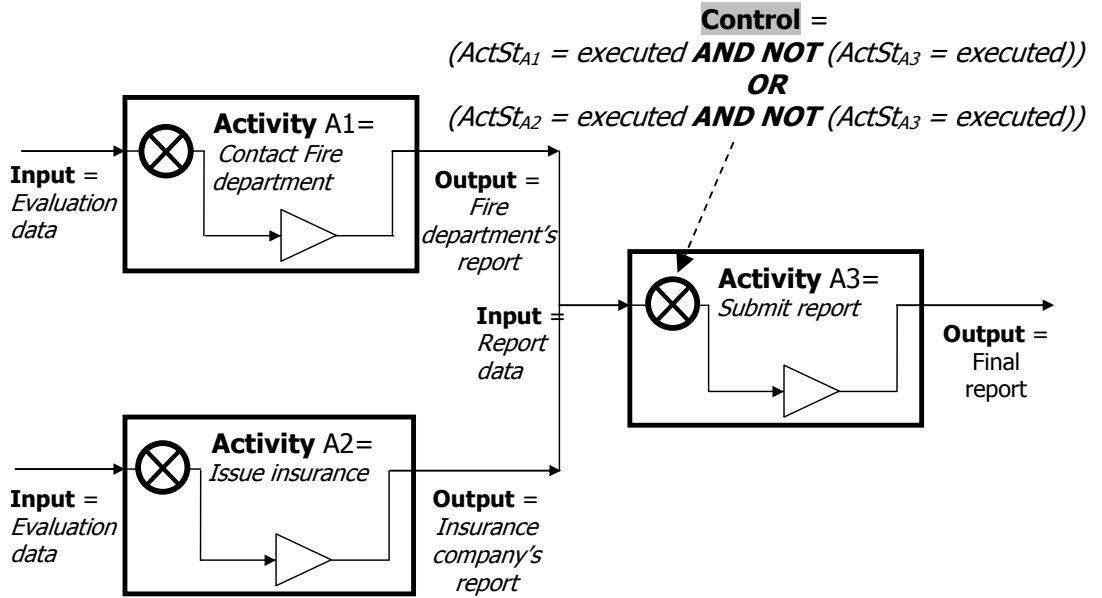


Figure 3.13: Workflow pattern 7: Synchronizing Merge

$$\begin{aligned}
 &\exists A_1, A_2, A_3 \in \Sigma : \\
 &\quad (I_{A1}(\text{Evaluation data}) \in I(A_1) \wedge O_{A1}(\text{Fire dept's report}) \in O(A_1)) \wedge \\
 &\quad (I_{A2}(\text{Evaluation data}) \in I(A_2) \wedge O_{A2}(\text{Insurance company's data}) \in O(A_2)) \wedge \\
 &\quad (I_{A3}(\text{Report data}) \in I(A_3) \wedge O_{A3}(\text{Final report}) \in O(A_3) \wedge C_{A3} \in C(A_3)) \wedge \\
 &\quad (I_{A3}(\text{Report data}) = O_{A1}(\text{Fire dept's report}) \vee O_{A2}(\text{Insurance company's data})) \wedge \\
 &\quad (C_{A3} = f: ((ActSt_{A1}=\text{executed} \text{ AND NOT } (ActSt_{A3}=\text{executed})) \text{ OR } \\
 &\quad \quad (ActSt_{A2}=\text{executed} \text{ AND NOT } (ActSt_{A3}=\text{executed}))) \rightarrow \\
 &\quad \rightarrow \{\text{True} \vee \text{False}\})
 \end{aligned}$$

Formula 3.23: Semantics of Workflow pattern 7: Synchronizing Merge

WP8 Multi-Merge: A point in a process where two or more branches merge without synchronization. If more than one branch gets activated, possibly concurrently, the activity following the merge is started for every action of every incoming branch.

Example: Sometimes two or more branches share the same ending. Two Activities “*Audit application*” and “*Process application*” are running in parallel, which should both be followed by an Activity “*Close case*”, which should be executed twice if the Activities “*Audit application*” and “*Process application*” are both executed.

Solution, WP8 The solution is depicted in Figure 3.14.

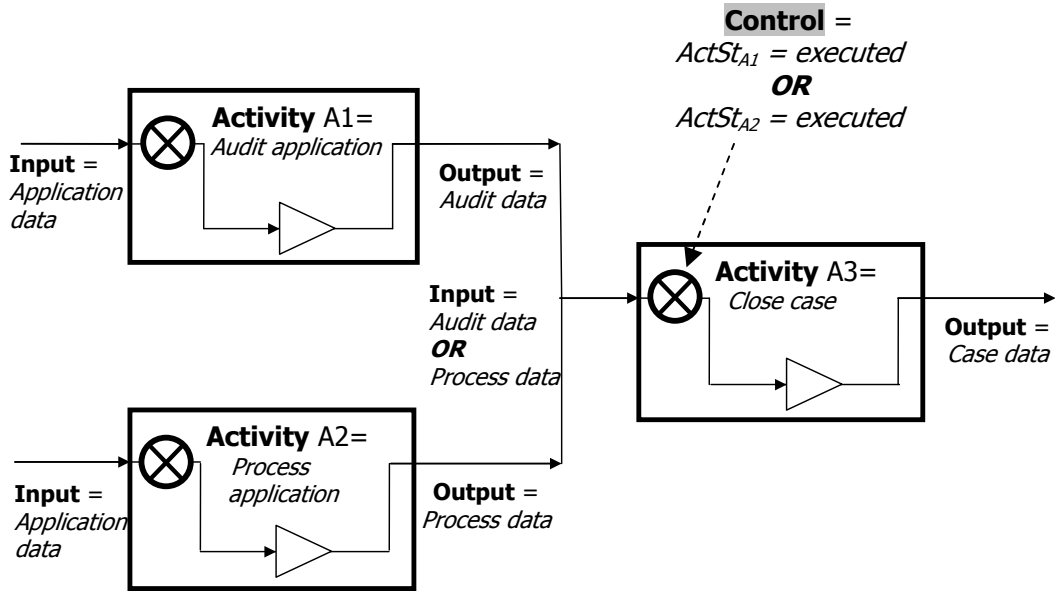


Figure 3.14: Workflow pattern 8: Multi-Merge

$\exists A_1, A_2, A_3 \in \Sigma :$

$$\begin{aligned}
 & (I_{A1}(\text{Application data}) \in I(A_1) \wedge O_{A1}(\text{Audit data}) \in O(A_1)) \wedge \\
 & (I_{A2}(\text{Application data}) \in I(A_2) \wedge O_{A2}(\text{Process data}) \in O(A_2)) \wedge \\
 & (I_{A3}(\text{Audit OR Process data}) \in I(A_3) \wedge O_{A3}(\text{Case data}) \in O(A_3) \wedge C_{A3} \in C(A_3)) \wedge \\
 & (I_{A3}(\text{Audit OR Process data}) = O_{A1}(\text{Audit data}) \vee O_{A2}(\text{Process data})) \wedge \\
 & (C_{A3} = f : (\text{ActSt}_{A1}=\text{executed OR ActSt}_{A2}=\text{executed}) \rightarrow \{\text{True} \vee \text{False}\})
 \end{aligned}$$

Formula 3.24: Semantics of Workflow pattern 8: Multi-Merge

WP9 Discriminator: A point in the workflow process that waits for one of the incoming branches to complete before activating the subsequent activity. From that moment on it waits for all remaining branches to complete and 'ignores' them. Once all incoming branches have been triggered, it resets itself so that it can be triggered again (which is important otherwise it could not really be used in the context of a loop).

Example: To improve query response time, a complex search is sent to two different databases over the Internet. The results of the first are used while the results of the second database are ignored.

Solution, WP9: Our formalism supports this pattern as it can employ **XOR** in Control of an Activity. The solution is depicted in Figure 3.15.

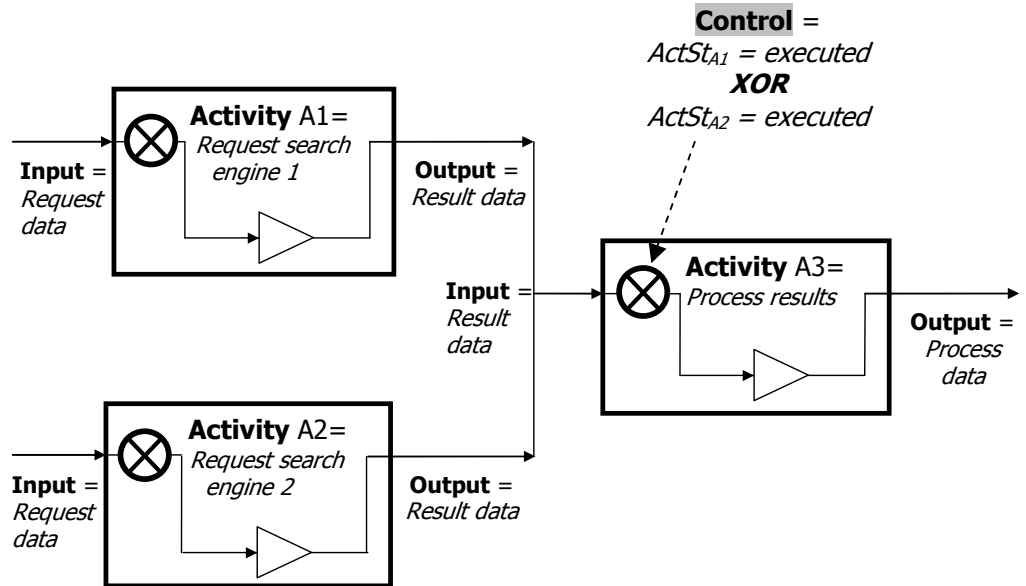


Figure 3.15: Workflow pattern 9: Discriminator

$\exists A_1, A_2, A_3 \in \Sigma :$

$$\begin{aligned}
 & (I_{A1}(\text{Request data}) \in I(A_1) \wedge O_{A1}(\text{Result data}) \in O(A_1)) \wedge \\
 & (I_{A2}(\text{Request data}) \in I(A_2) \wedge O_{A2}(\text{Result data}) \in O(A_2)) \wedge \\
 & (I_{A3}(\text{Result data}) \in I(A_3) \wedge O_{A3}(\text{Process data}) \in O(A_3) \wedge C_{A3} \in C(A_3)) \wedge \\
 & (I_{A3}(\text{Result data}) = O_{A1}(\text{Result data}) \vee O_{A2}(\text{Result data})) \wedge \\
 & (C_{A3} = f : (ActSt_{A1}=executed \text{ XOR } ActSt_{A2}=executed) \rightarrow \{\text{True} \vee \text{False}\})
 \end{aligned}$$

Formula 3.25: Semantics of Workflow pattern 9: Discriminator

WP10 Arbitrary Cycles: A point where a portion of the process (including one or more activities and connectors) needs to be "visited" repeatedly without imposing restrictions on the number, location, and nesting of these points.

Solution, WP10: This pattern is not directly supported by our formalism.

WP11 Implicit Termination: A given sub process is terminated when there is nothing left to do, i.e., termination does not require an explicit termination activity.

Solution, WP11: Implicit termination is supported by our formalism, as a process completes when its final Activity completes and there are no other Activities left for execution.

WP12 MI without Synchronization: Within the context of a single case multiple instances of an activity may be created. The instances might be created consecutively, but they will be able to run in parallel, which distinguishes this pattern from the pattern for Arbitrary Cycles.

Example: When booking a trip, the activity "*Book flight*" is executed multiple times if the trip involves multiple flights.

Solution, WP12 Multiple instances of an activity can be created by using the **Control** concept.

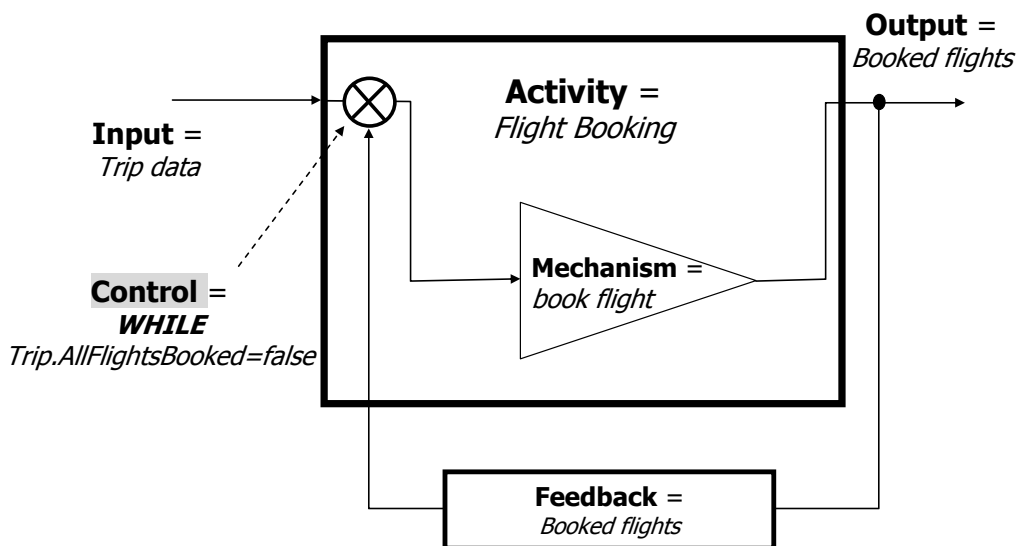


Figure 3.16: Workflow pattern 12: MI without Synchronization

WP13-WP15 MI with Synchronization A point in a workflow where a number of instances of a given activity are initiated, and these instances are later synchronized, before proceeding with the rest of the process. In WP13, the number of instances to be started/synchronized is known at design time. In WP14, the number is known at some stage during run time, but before the initiation of the instances has started. In WP15 the number of instances to be created is not known in advance: new instances are created on demand, until no more instances are required.

Example of WP15: When booking a trip, the activity “*Book flight*” is executed multiple times if the trip involves multiple flights. Once all bookings are made, an invoice is sent to the client. The number of the bookings is only known at runtime through interaction with the user.

Solutions, WP13-WP15 A simple solution, in case the number of instances to be synchronized is known at design time (WP13), is to replicate the activity as many times as it needs to be instantiated, and run the replicas in parallel by placing them in parallel activities (Figure 3.17 – we need to book 2 flights). If the number of instances to be created and synchronized is only known at run time (WP14), or not known (WP15) we may encode this by means of a loop encoded in Control concept (

Figure 3.18).

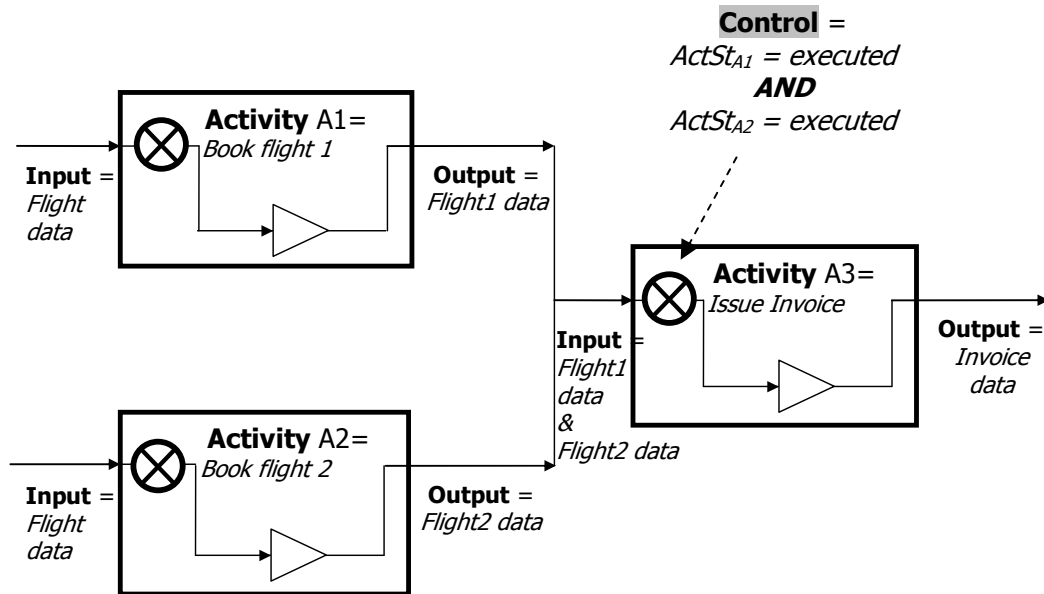
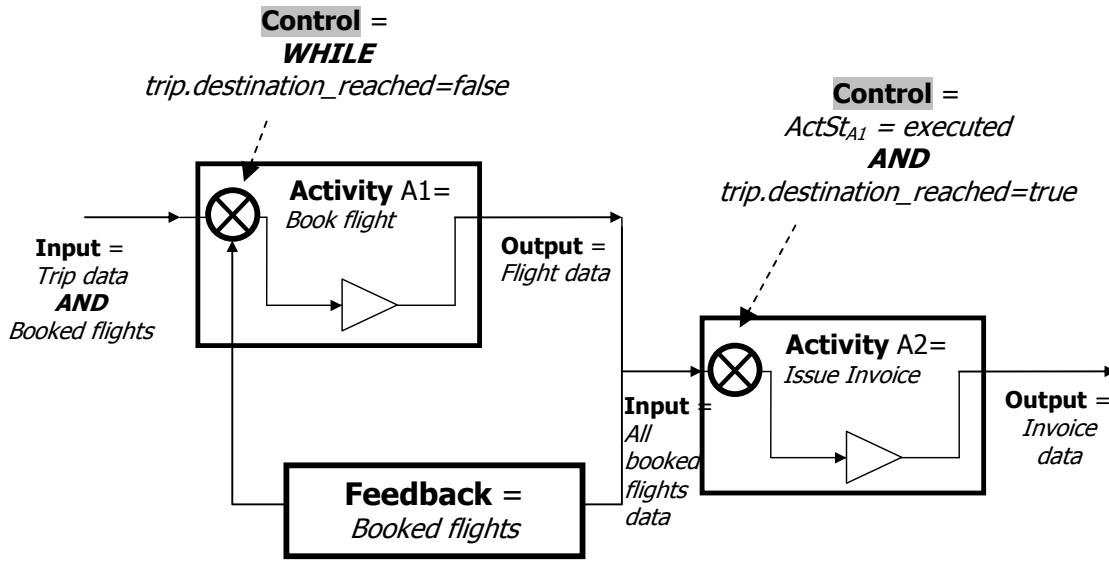


Figure 3.17: Workflow pattern 13: MI with Synchronization (a)

$$\exists A_1, A_2, A_3 \in \Sigma :$$

$$\begin{aligned} & (I_{A1}(\text{Flight data}) \in I(A_1) \wedge O_{A1}(\text{Flight1 data}) \in O(A_1)) \wedge \\ & (I_{A2}(\text{Flight data}) \in I(A_2) \wedge O_{A2}(\text{Flight2 data}) \in O(A_2)) \wedge \\ & (I_{A3}(\text{Flight1 \& Flight2 data}) \in I(A_3) \wedge O_{A3}(\text{Invoice data}) \in O(A_3) \wedge C_{A3} \in C(A_3)) \wedge \\ & (I_{A3}(\text{Flight1 \& Flight2 data}) = O_{A1}(\text{Flight1 data}) \wedge O_{A2}(\text{Flight2 data})) \wedge \\ & (C_{A3} = f : (\text{ActSt}_{A1}=\text{executed AND } \text{ActSt}_{A2}=\text{executed}) \rightarrow \{\text{True} \vee \text{False}\}) \end{aligned}$$

Formula 3.26: Semantics of Workflow pattern 14: MI with Synchronization (a)

Figure 3.18: Workflow pattern 14-15: MI with Synchronization (b)

$$\exists A_1, A_2 \in \Sigma :$$

$$\begin{aligned} & (I_{A1}(\text{Trip data \& Booked flights}) \in I(A_1) \wedge O_{A1}(\text{Flight1 data}) \in O(A_1) \wedge F_{A1} \in F(A_1)) \wedge \\ & (I_{A2}(\text{All booked flight data}) \in I(A_2) \wedge O_{A2}(\text{Invoice data}) \in O(A_2) \wedge C_{A2} \in C(A_2)) \wedge \\ & (F_{A1} \subseteq O_{A1}(\text{Flight1 data})) \wedge \\ & (I_{A1}(\text{Trip data \& Booked flights}) = I_{A1}(\text{Trip data \& Booked flights}) \wedge F_{A1}) \wedge \\ & (C_{A1} = f : (\text{WHILE } \text{trip.destination_reached=false}) \rightarrow \{\text{True} \vee \text{False}\}) \wedge \\ & (C_{A2} = f : (\text{ActSt}_{A1}=\text{executed AND } \text{trip.destination_reached=true}) \rightarrow \{\text{True} \vee \text{False}\}) \end{aligned}$$

Formula 3.27: Semantics of Workflow pattern 14-15: MI with Synchronization (b)

WP16 Deferred Choice: A point in a process where one among several alternative branches is chosen based on information, which is not necessarily available when this point is reached. This differs from the normal exclusive choice, in that the choice is not made immediately when the point is reached, but instead several alternatives are offered, and the choice between them is delayed until the occurrence of some event.

Example: When a contract is finalized, it has to be reviewed and signed either by the director or by the operations manager, whoever is available first. Both the director and the operations manager would be notified that the contract is to be reviewed: the first one who is available will proceed with the review.

Solution, WP16: The solution is depicted in Figure 3.19.

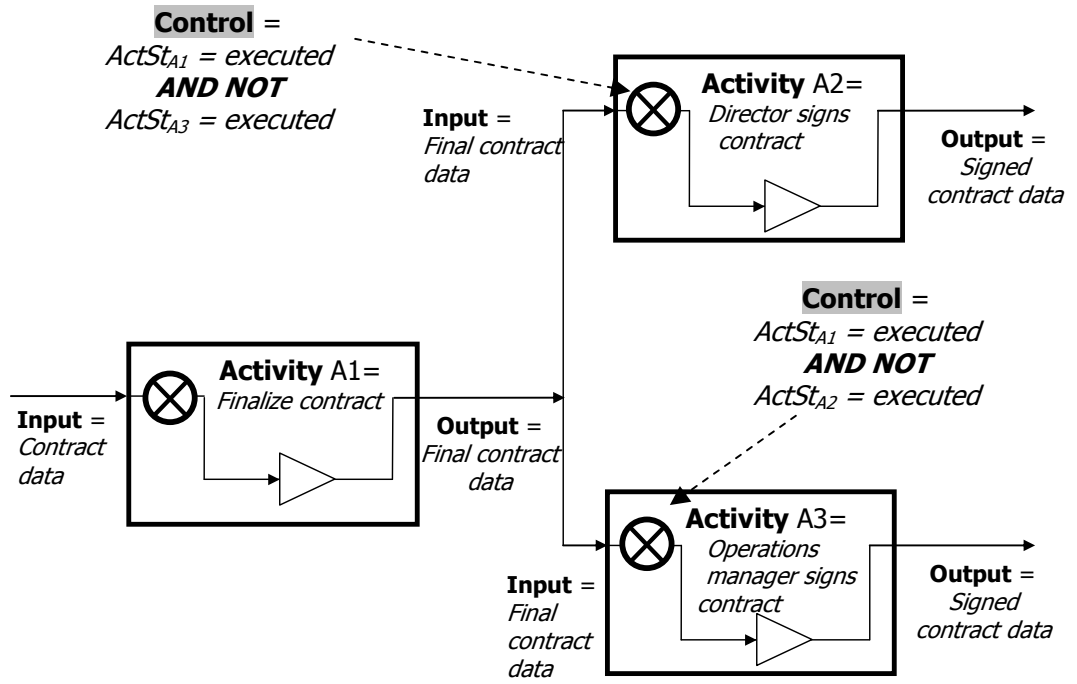


Figure 3.19: Workflow pattern 16: Deferred Choice

$\exists A_1, A_2, A_3 \in \Sigma :$

$$\begin{aligned}
 & (I_{A1}(\text{Contract data}) \in I(A_1) \wedge O_{A1}(\text{Final contract data}) \in O(A_1)) \wedge \\
 & (I_{A2}(\text{Final contract data}) \in I(A_2) \wedge O_{A2}(\text{Signed contract data}) \in O(A_2) \wedge C_{A2} \in C(A_2)) \wedge \\
 & (I_{A3}(\text{Final contract data}) \in I(A_3) \wedge O_{A3}(\text{Signed contract data}) \in O(A_3) \wedge C_{A3} \in C(A_3)) \wedge \\
 & (I_{A2}(\text{Final contract data}) = I_{A3}(\text{Final contract data}) = O_{A1}(\text{Final contract data})) \wedge \\
 & (C_{A2} = f : (\text{ActSt}_{A1}=\text{executed AND NOT ActSt}_{A3}=\text{executed}) \rightarrow \{\text{True} \vee \text{False}\}) \wedge \\
 & (C_{A3} = f : (\text{ActSt}_{A1}=\text{executed AND NOT ActSt}_{A2}=\text{executed}) \rightarrow \{\text{True} \vee \text{False}\})
 \end{aligned}$$

Formula 3.28: Semantics of Workflow pattern 16: Multi-Choice

WP17 Interleaved Parallel Routing: A set of activities is executed in an arbitrary order. Each activity in the set is executed exactly once. The order between the activities is decided at run-time: it is not until one activity is completed that the decision on what to do next is taken. In any case, no two activities in the set can be active at the same time.

Example: At the end of each year, a bank executes two activities for each account: "*Add interest*" and "*Charge credit card costs*". These activities can be executed in any order. However, since they both update the account, they cannot be executed at the same time.

Solution, WP17: Our formalism does not have a construct like BPEL4WS' concept of serializable scopes.

WP18 Milestone: A given activity can only be executed if a certain milestone has been reached which has not yet expired. A milestone is defined as a point in the process where a given activity A has finished and an activity B following it has not yet started.

Example: After having placed a purchase order, a customer can withdraw it at any time before the shipping takes place. To withdraw an order, the customer must complete a withdrawal request form, and this request must be approved by a customer service representative. The execution of Activity "*Approve order withdrawal*" must therefore follow Activity "*Request withdrawal*", only if: (i) the activity "*Place order*" is completed, and (ii) the Activity "*Ship order*" has not yet started.

Solution, WP18: Our formalism does not provide direct support for this pattern.

WP19 Cancel Activity & WP20 Cancel Case: A cancel activity terminates a running instance of an activity, while cancelling a case leads to the removal of an entire workflow instance.

Example of WP19: A customer cancels a request for information.

Example of WP20: A customer withdraws his/her order.

Solutions, WP19 & WP20: WP20 is not supported by our formalism while WP19 is dealt with using compensation and faults activities, specifying the course of action in cases of faults and cancellations.

In this Section, we made an in-depth analysis of SYMEX based on existing workflow patterns. A summary of the results from the analysis are presented in Table 3.2. The table also shows a comparison of our framework with BPEL4WS, BPMN, and UML Activity Diagrams. A '+' in a cell of the table refers to direct support (i.e. there is a construct in the language which directly support the pattern). A '-' in the table refers to no direct support. Sometimes there is a feature that only partially supports a pattern, e.g., a construct that implies certain restrictions on the structure of the process. In such cases, the support is rated as '+/-'.

	<i>WF pattern Description</i>	<i>product/standard</i>			
		BPEL4WS	BPMN	UML	SYMEX
WP1	Sequence	+	+	+	+
WP2	Parallel Split	+	+	+	+
WP3	Synchronization	+	+	+	+
WP4	Exclusive Choice	+	+	+	+
WP5	Simple Merge	+	+	+	+
WP6	Multi Choice	+	+	+	+
WP7	Synchronizing Merge	+	+	-	+
WP8	Multi Merge	-	+	+	+
WP9	Discriminator	-	+/-	+/-	+
WP10	Arbitrary Cycles	-	+	+	+/-
WP11	Implicit Termination	+	+	+	+
WP12	MI without Synchronization	+	+	+	+
WP13	MI with a Priori Design Time Knowledge	+	+	+	+
WP14	MI with a Priori Runtime Knowledge	-	+	+	+
WP15	MI without a Priori Runtime Knowledge	-	-	-	+
WP16	Deferred Choice	+	+	+	+
WP17	Interleaved Parallel Routing	+	-	-	-
WP18	Milestone	-	-	-	-
WP19	Cancel Activity	+	+	+	+
WP20	Cancel Case	+	+	+	-

Table 3.2: Comparison based on workflow patterns [176]

The following observations can now be made from the table:

1. The first five patterns correspond to the basic routing constructs and they are naturally supported by all languages/notations. In contrast, some of the patterns referring to more advanced constructs are often not supported in the different approaches, e.g. WP8, WP9, WP10 are not supported by BPEL4WS, WP15 is only supported by SYMEX etc.

2. SYMEX supports an adequate number of workflow patterns and thus can be considered as a competent approach for capturing and encoding low-level workflow execution semantics.
3. While other approaches use control flow constructs to model workflow patterns, SYMEX modelling does not realize patterns using modelling constructs. Instead, SYMEX models use the construct of Controls of Activities in order to constrain their execution and implicitly realize the desired workflow pattern.

3.5 Overall comparison

In the previous Section, we examined the capability of the proposed framework to capture the business semantics identified in high-level workflow process descriptions and to support low-level workflow execution semantics in terms of capturing workflow patterns that encode typical control flow of workflow execution. This Section discusses the results of this analysis and makes an overall comparison of SYMEX with UML and BPMN as the most widely adopted workflow process modelling languages and with BPEL4WS as the most widely adopted process execution framework.

Table 3.3 below combines the results of Sections 3.4.1 and 3.4.2, where we examined the support of high-level business semantics and low-level execution patterns as set in Section 3.2. A '+' in a cell of the table refers to direct support, a '-' to no direct support and a '+/-' to partial support but with certain restrictions.

	UML 2 AD	BPMN	BPEL4WS	SYMEX
Number of basic constructs	13	11	4	5
Support of high-level semantics				
Flow representation	+	+	+	+
Conditions and effects	+	+	+	+
Role assignment	+	+	+	+
Hierarchy representation	+	+	-	+
Feedback loops	-	+/-	-	+
Support of low-level workflow patterns (fully and partially supported)	16	17	14	17
Degree of efficiency (supported semantics+patterns divided by the number of basic constructs)	1,54	1,95	4,25	4,4

Table 3.3: Overall comparison of SYMEX to other approaches

Based on the aggregated results of Table 3.3 we can make the following claims:

1. SYMEX proves to be capable of capturing all aspects of high-level business descriptions that we have set as minimum requirements in Section 3.2.
2. SYMEX proves to support 17 workflow patterns and together with BPMN are the two most efficient approaches for encoding low-level workflow execution.
3. SYMEX has five (5) basic constructs and comparing its capability of supporting high and low-level descriptions of workflow processes, makes it a more concise and at the same time more efficient approach than the others.

3.6 Conclusions

One of the main goals of workflow process modelling is to establish precise structure–function relationships between the dynamics of a business environment and computer based execution of business activities. This was traditionally accomplished by manually translating high-level descriptions of process models to low-level code for workflow execution purposes. Only recently reported research has proposed some mappings between high-level descriptions and low-level execution semantics, but this research is still in early stages and faces an number of issues, as discussed in Chapter 2.

In order to address our first research objective of integrating high-level with low-level execution semantics, we have defined a number of requirements for an integrated approach in Section 3.2. Then, inspired by Systems Theory we visualized a workflow model as an entity and the two models (high-level and low-level) as the elements of this entity. In addition, the concept of hierarchy proposed by Systems Theory helped us to define hierarchical models where the upper layers are depicting high-level descriptions and lower-layers encode the execution semantics. The result was the definition of SYMEX, a Systems Theory based framework that encodes workflow processes, as a set of Activities taking Inputs and producing Outputs. Activities are executed via the means of their execution Mechanisms and are activated by their Control. The mathematical descriptions of the concepts were also defined in order to provide a sound foundation for automatic validation and computer-based execution of the modelled workflow processes.

After the theory, we have made an analysis of the framework in terms of its capability of capturing and representing high-level descriptions and low-level workflow processes in an integrated way. SYMEX proved to comply with the set of requirements we have defined in Section 3.2 and the comparison with other workflow modelling approaches in Section 3.5 showed the advantage of the framework in terms of its degree of efficiency in modelling workflow processes. However, for further assessing the applicability of SYMEX in business process scenarios, we need to provide a mechanism for executing the integrated workflow process models. This is the main objective of the next Chapter.

4 Application of SYMEX

4.1 Introduction

In Chapter 3, we introduced our theoretical framework, SYMEX, in terms of its main concepts and their expressiveness capability. We then analysed and discussed how SYMEX integrates business semantics captured in high-level descriptions and workflow execution patterns encoded in low-level descriptions. However, for assessing further the applicability of SYMEX in business process scenarios, this Chapter proposes an approach to develop a prototype of SYMEX and present an implemented workflow execution engine that executes SYMEX models.

Specifically, we propose an XML-based implementation approach in order to design and develop a working prototype of SYMEX. We choose XML because it can be processed easily by computers, and it is an extensible and open cross-platform W3C standard, endorsed by software industry market leaders [178-181]. This Chapter also demonstrates how constraint-based reasoning adopted by SYMEX models can facilitate and accelerate adaptation of processes in comparison with approaches that explicitly sequence activities of processes. Finally, we present the workflow inference engine we have implemented in order to execute SYMEX models and discuss the Information System modelled with SYMEX.

The structure of this Chapter is as follows. Section 4.2 presents our XML-based implementation approach. Section 4.3 applies SYMEX modelling on a business process derived from the accounting domain. Then, Section 4.4 makes a comparative analysis of change management between a SYMEX model and a UML Activity Diagram model. Section 4.5 describes the constraint-based algorithm of our implemented workflow inference engine. Section 4.6 discusses the application of SYMEX for an Information System and finally, Section 4.7 concludes the Chapter.

4.2 Process modelling

In Chapter 3, we have formally defined the concepts of our framework, their properties, and relationships. This Section encodes this logic using XML technology, defining the concepts of SYMEX and the relationships between them in a top-down approach. At a top level, we have the concept of Process. So beginning by the definition of a Process, we describe its detailed elements.

4.2.1 Process

A business process is defined as an element of type **Process** having the following *attributes*:

1. **ID**, which is the unique identifier of the process
2. **Name**, which is a descriptive name of the process
3. **Codename**, which is the codename used by the execution engine (see Section 4.5)
4. **Description**, which describes the process.

The *elements* of the **Process** are:

1. **Entity Pool** that includes elements of types **Schemas**, **Controls**, and **Mechanisms**.
2. **Activities** that includes all the elements of type **Activity**.

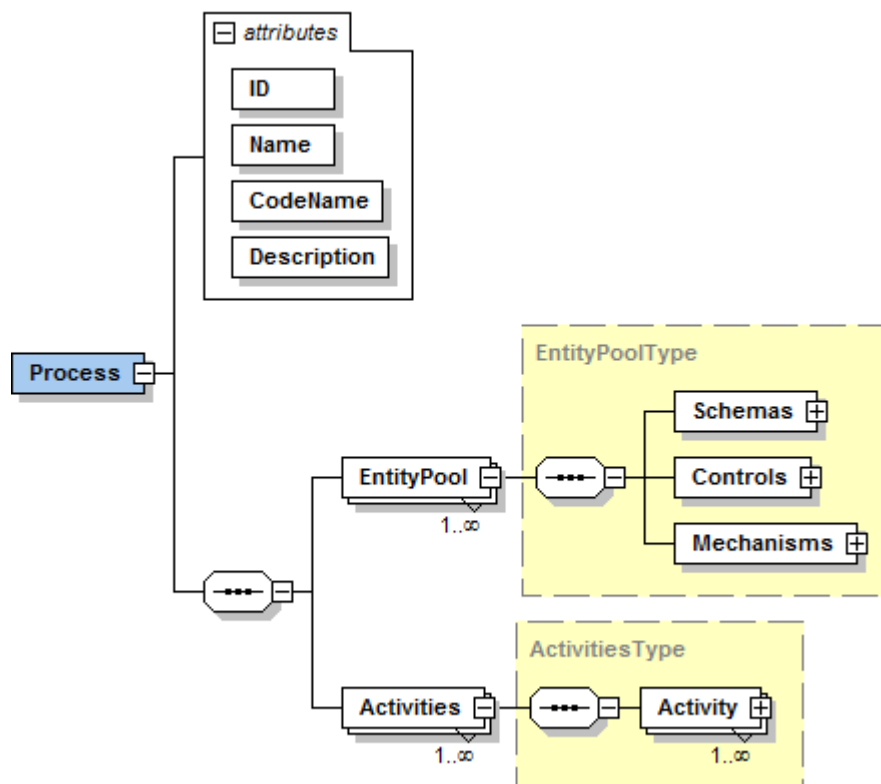


Figure 4.1: Process: attributes and elements

4.2.2 Entity Pool

4.2.2.1 Schemas

Schemas is an element including all the elements of type **Schema**. Element of type **Schema** is defined inside the **Entity Pool** of the **Process**, as there may be **Activities** with common **Inputs** and **Outputs** inside the **Process**. Thus, **Inputs** and **Outputs** of **Activities** are of type **Schema**,

Schema has one *attribute*:

1. **ID**, which is the unique identifier of the Schema

The *elements* of the **Schema** are:

1. **Name**, which is a descriptive name of the Schema
2. **Description**, which describes the Schema
3. **XSD**, which is an XML Schema Definition describing the structure of the Schema. A valid instance of a Schema is essentially an XML validated by its XSD.

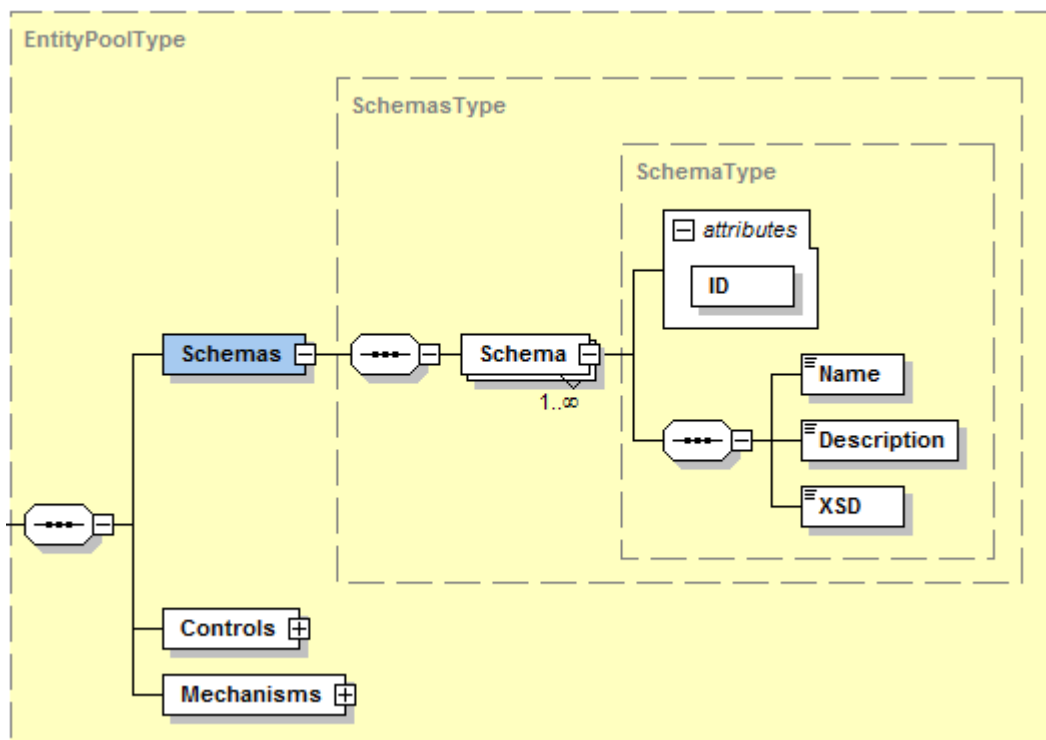


Figure 4.2: Schema: attributes and elements

4.2.2.2 Controls

Controls is an element including all the elements of type **Control**. Element of type **Control** is defined inside the **Entity Pool** of the **Process**, as there may be **Activities** with common **Controls** inside the **Process**.

Control has one *attribute*:

1. **ID**, which is the unique identifier of the Control

The *elements* of the **Control** are:

1. **Name**, which is a descriptive name of the Control
2. **Codename**, which is the codename used by the execution engine (see Section 4.5)
3. **Condition**, which is a Boolean expression containing states of Activities of the process and data from elements of **Schema** (**Inputs**).

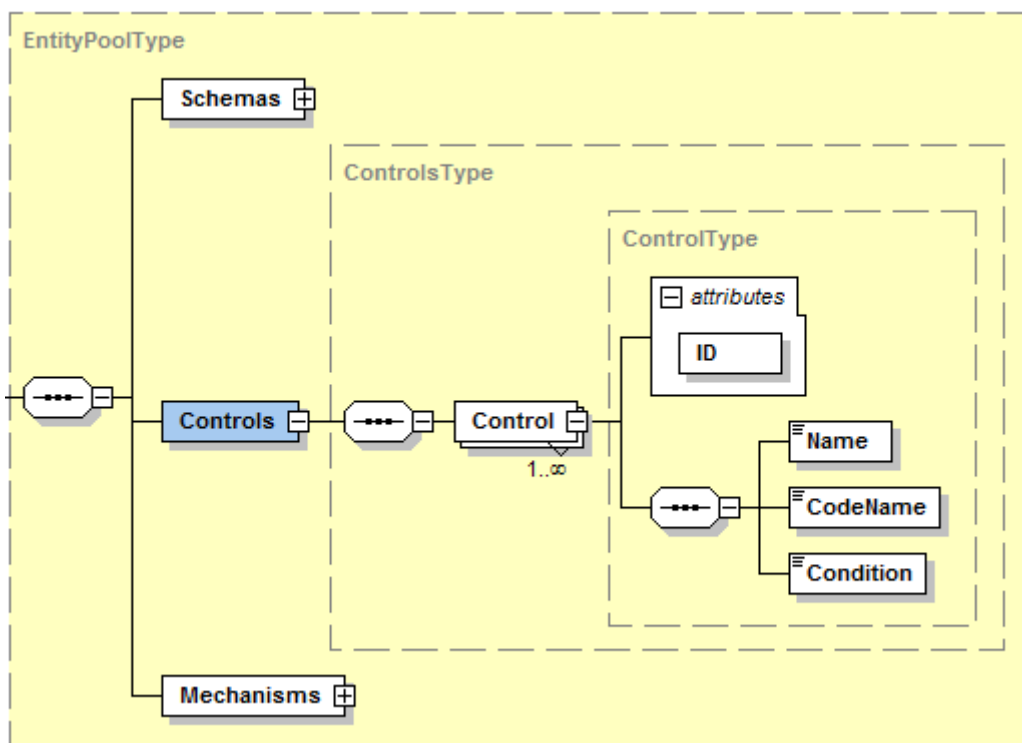


Figure 4.3: Control: attributes and elements

4.2.2.3 Mechanisms

Mechanisms is an element including all the elements of type **Mechanism**. Element of type **Mechanism** is defined inside the **Entity Pool** of the **Process**, as there may be **Activities** with common **Mechanisms** inside the **Process**.

Mechanism has one *attribute*:

1. **ID**, which is the unique identifier of the Mechanism

The *elements* of the **Mechanism** are:

1. **Name**, which is a descriptive name of the Mechanism
2. **Description**, which describes the Mechanism
3. **Type**. There are 3 different types of Mechanism:
 - a) **Role**; this type of Mechanism defines one or more roles. When this Mechanism is applied to an Activity, only the roles defined in the Mechanism can enact the Activity. In this case the element **Roles** of Mechanism is used to define the available roles in XML format.
 - b) **Component**; this type of Mechanism defines execution of a software component. When this Mechanism is applied to an Activity, the defined component is executed. In this case the element:
 1. **ProgID** is used to define the Programmatic Id of the software component, e.g. ExampleObject.dll,
 2. **Method** is used to define the invoked method of the software component, e.g. CalculateTax() and
 3. **Arguments** are used to define the input parameters of the software component, e.g. Invoice.TotalAmount. **Arguments** use subset of the Input elements of Activities.

After the execution of the software component, the produced output parameter is set as **Output** (element of type **Schema**) of the **Activity**.

- c) **XSLT**; this type of Mechanism defines a transformation on the XML structure containing the **Input** (element of type **Schema**) of the **Activity**. In this case the element **XSLT** is used to define the XSL Transformation. The result of the transformation is either set as input parameter to a Mechanism of type Component (if there is one available) or is directly set as **Output** (element of type **Schema**) of the **Activity**.

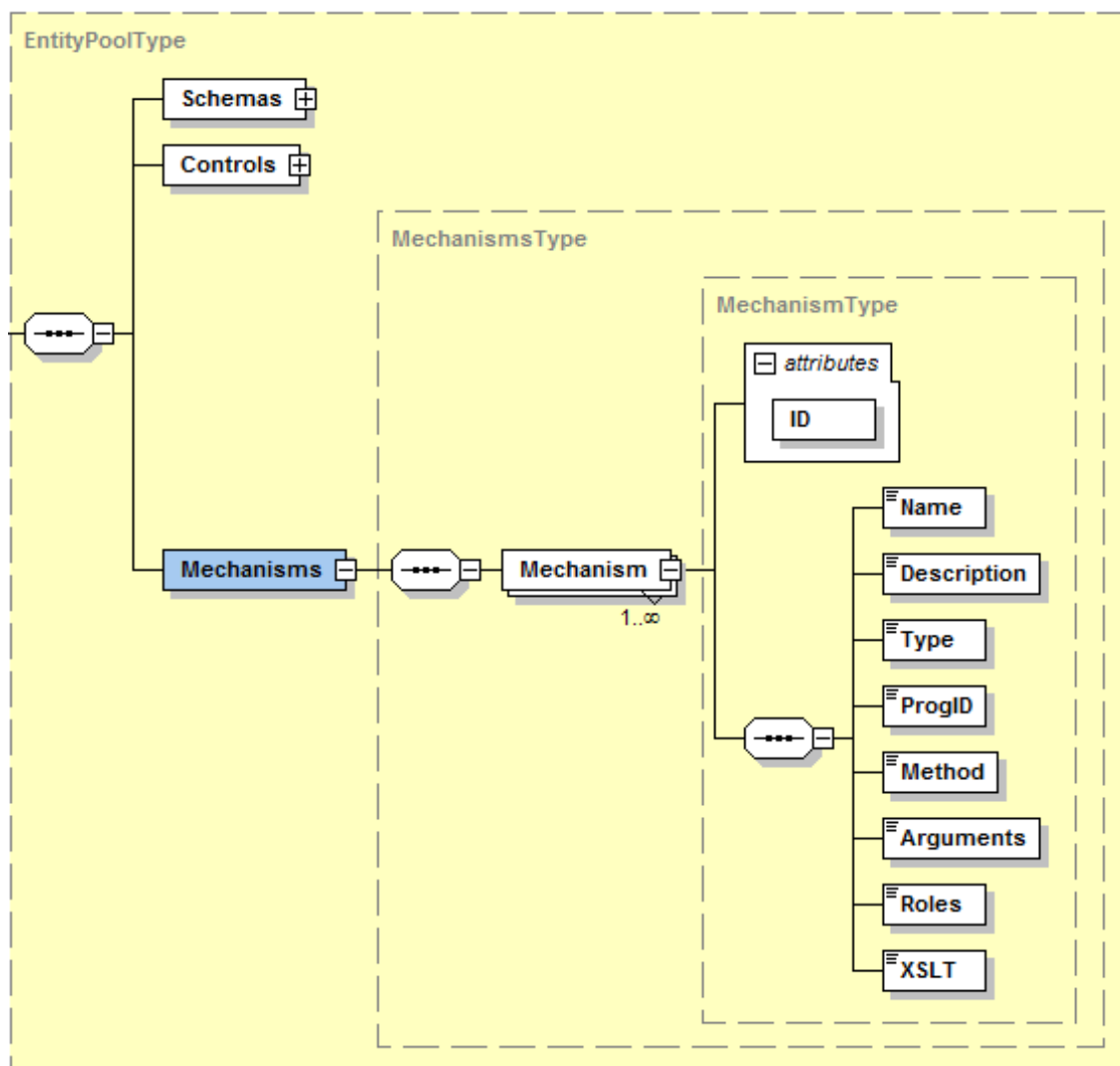


Figure 4.4: Mechanism: attributes and elements

4.2.3 Activities

Activities is an element including all the elements of type **Activity**.

Activity has the following *attributes*:

1. **ID**, which is the unique identifier of the Activity
2. **Name**, which is a descriptive name of the Activity
3. **CodeName**, which is the codename used by the execution engine (see Section 4.5)
4. **Description**, which describes the Activity
5. **PendingJobInfo**, which describes in a XML-based format the information that will be exposed to the user when an Activity waits for human intervention. **PendingJobInfo** uses data from the **Input** (element of type **Schema**) of the **Activity**.
6. **ControlSynch**, which is a Boolean expression containing data from elements of type **Control** of the **Activity**.

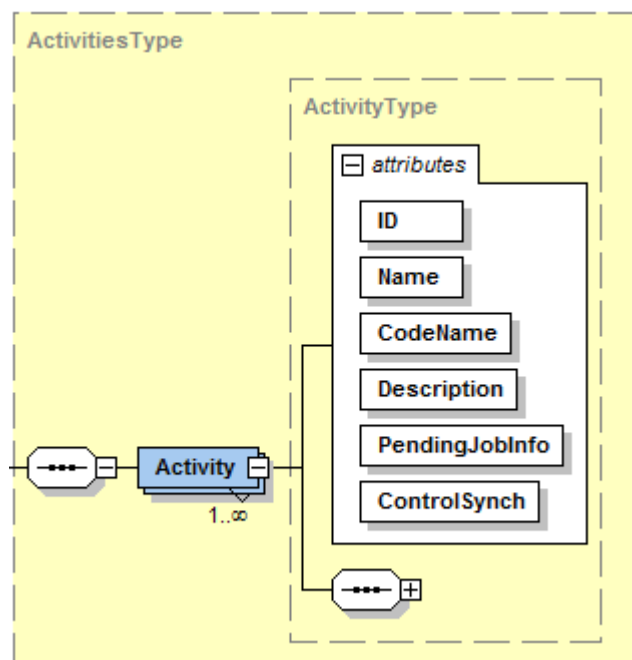


Figure 4.5: Activity: attributes

An **Activity** has the following *elements*:

1. **Inputs**, which are of type **Schema** and use the reference **ID** to resolve an exact **Schema** from the **Entity Pool**.
2. **Outputs**, which are of type **Schema** and use the reference **ID** to resolve an exact **Schema** from the **Entity Pool**. **Outputs** have also one attribute: **RuntimeMainOutput**; this is used for execution purposes, as the current implementation of our workflow inference engine (see Section 4.5), assumes that an **Activity** can produce only one **Output**, while SYMEX models allow for multiple outputs.

3. **Controls**, which are of type **Control** and use the reference **ID** to resolve an exact **Control** from the **Entity Pool**.
4. **Mechanisms**, which are of type **Mechanism** and use the reference **ID** to resolve an exact **Mechanism** from the **Entity Pool**.

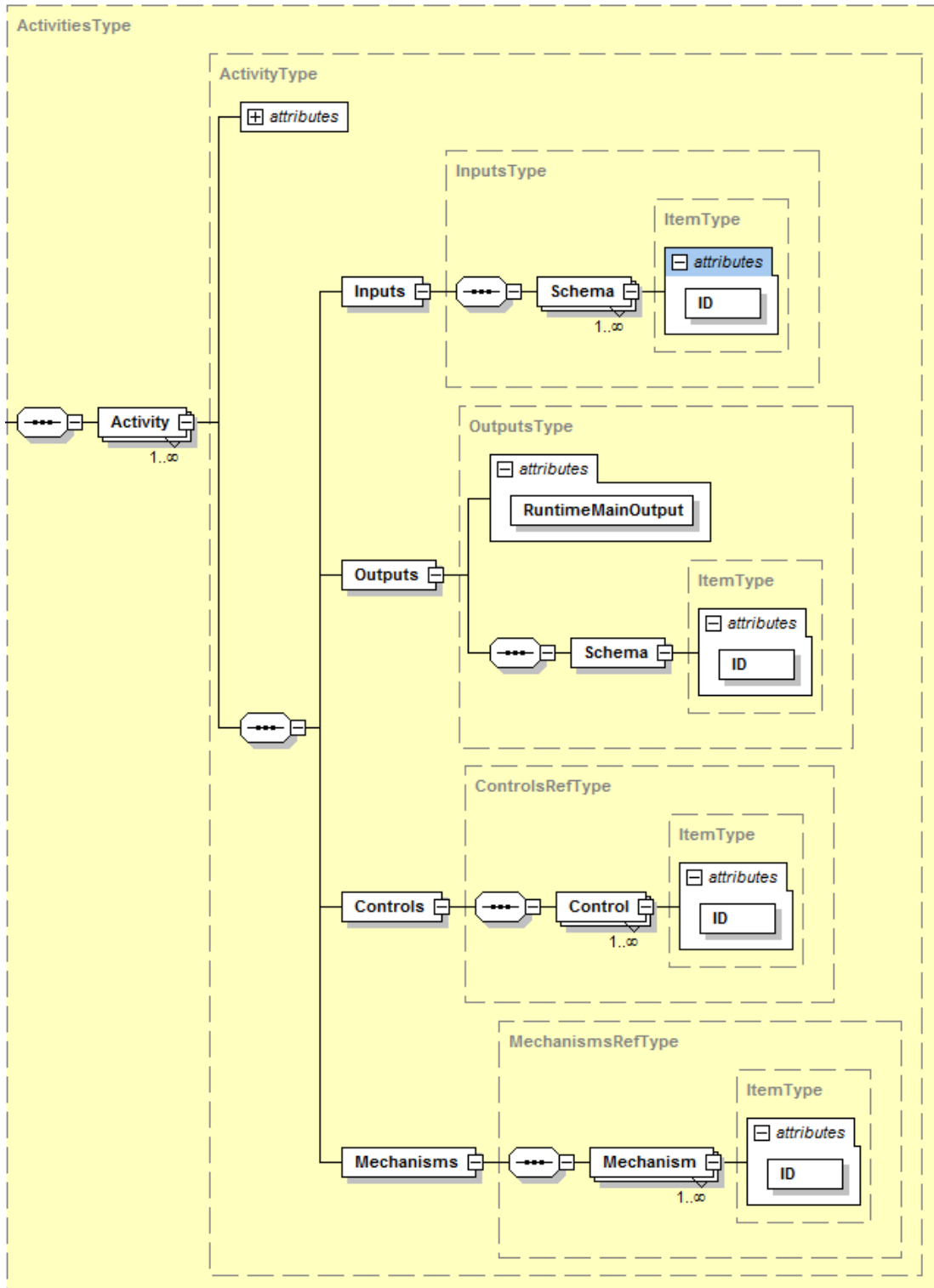


Figure 4.6: Activity: elements

4.2.4 XML Schema Definition

Figure 4.7 presents the entire XML Schema Definition (XSD) that captures SYMEX constructs.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:complexType name="ActivitiesType">
    <xs:sequence>
      <xs:element name="Activity" type="ActivityType" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ActivityType">
    <xs:sequence>
      <xs:element name="Inputs" type="InputsType"/>
      <xs:element name="Outputs" type="OutputsType"/>
      <xs:element name="Controls" type="ControlsRefType"/>
      <xs:element name="Mechanisms" type="MechanismsRefType"/>
    </xs:sequence>
    <xs:attribute name="ID" type="xs:string" use="required"/>
    <xs:attribute name="Name" type="xs:string" use="required"/>
    <xs:attribute name="CodeName" type="xs:string" use="required"/>
    <xs:attribute name="Description" type="xs:string" use="required"/>
    <xs:attribute name="PendingJobInfo" type="xs:string" use="required"/>
    <xs:attribute name="ControlSynch" type="xs:string" use="required"/>
  </xs:complexType>
  <xs:complexType name="ControlType">
    <xs:sequence>
      <xs:element name="Name"/>
      <xs:element name="CodeName"/>
      <xs:element name="Condition"/>
    </xs:sequence>
    <xs:attribute name="ID" type="xs:string" use="required"/>
  </xs:complexType>
  <xs:complexType name="ControlsType">
    <xs:sequence>
      <xs:element name="Control" type="ControlType" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ControlsRefType">
    <xs:sequence>
      <xs:element name="Control" type="ItemType" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="InputsType">
    <xs:sequence>
      <xs:element name="Schema" type="ItemType" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ItemType">
    <xs:attribute name="ID" type="xs:string" use="required"/>
  </xs:complexType>
  <xs:complexType name="MechanismType">
    <xs:sequence>
      <xs:element name="Name"/>
      <xs:element name="Description"/>
      <xs:element name="Type"/>
      <xs:element name="ProgID"/>
      <xs:element name="Method"/>
      <xs:element name="Arguments"/>
      <xs:element name="Roles"/>
      <xs:element name="XSLT"/>
    </xs:sequence>
    <xs:attribute name="ID" type="xs:string" use="required"/>
  </xs:complexType>
  <xs:complexType name="MechanismsType">
    <xs:sequence>
      <xs:element name="Mechanism" type="MechanismType" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="MechanismsRefType">
    <xs:sequence>
      <xs:element name="Mechanism" type="ItemType" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="OutputType">
    <xs:sequence>
      <xs:element name="Schema" type="ItemType"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="OutputsType">
    <xs:sequence>
      <xs:element name="Schema" type="ItemType"/>
    </xs:sequence>
    <xs:attribute name="RuntimeMainOutput" type="xs:string" use="required"/>
  </xs:complexType>
</xs:schema>
```

```

</xs:complexType>
<xs:element name="Process">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="EntityPool" type="EntityPoolType" maxOccurs="unbounded"/>
      <xs:element name="Activities" type="ActivitiesType" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="ID" type="xs:string" use="required"/>
    <xs:attribute name="Name" type="xs:string" use="required"/>
    <xs:attribute name="CodeName" type="xs:string" use="required"/>
    <xs:attribute name="Description" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
<xs:complexType name="SchemaType">
  <xs:sequence>
    <xs:element name="Name"/>
    <xs:element name="Description"/>
    <xs:element name="XSD"/>
  </xs:sequence>
  <xs:attribute name="ID" type="xs:string" use="required"/>
</xs:complexType>
<xs:complexType name="SchemasType">
  <xs:sequence>
    <xs:element name="Schema" type="SchemaType" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="EntityPoolType">
  <xs:sequence>
    <xs:element name="Schemas" type="SchemasType"/>
    <xs:element name="Controls" type="ControlsType"/>
    <xs:element name="Mechanisms" type="MechanismsType"/>
  </xs:sequence>
</xs:complexType>
</xs:schema>

```

Figure 4.7: XML Schema Definition for capturing SYMEX constructs

4.3 Application of SYMEX modelling

This Section applies SYMEX modelling on a business process derived from the accounting domain. First, the business process is described in natural language and then its semantics are captured using SYMEX approach. At the end of the Section, we present the corresponding process description in terms of the XML-based implementation we have proposed in the previous Section of this Chapter.

4.3.1 Process description

The business process we model is based on the process realized at the Accounting Office of Athens University of Economics and Business [182] that we recorded in September of 2008. Names in bold indicate names of **Activities**, **Roles**, **Inputs** and **Outputs** of the process.

Athens University of Economics and Business has an **Accounting Office** that is responsible for all the purchases of the various departments of the University. Whenever a department wants to purchase something, the **secretary** of the department makes a **Purchase Request** to the **Accounting Office**. The **Accounting Office registers** the **Purchase Request** and it either **approves** or **disapproves it**. If the **Purchase Request** is **approved**, the **secretary** of the department is **informed** and contacts the **Supplier** to **order** the **approved items**. The **Supplier** then **issues an Invoice** for the ordered items and the **Accounting Office receives** and **registers** the **Invoice**. Finally, when the **secretary** of the department **informs** the **Accounting Office** that the **items are delivered and checked**, the **Accounting Office issue a Payment Order** and a **Check** for the **Supplier**, based on the **approved Purchase Request** and the **registered Invoice**. The **Supplier** is given the **Check** and the **Accounting Office receives a Payment Receipt**.

4.3.2 SYMEX model

The SYMEX model of the example is composed of two levels:

1. The first level consists of three Activities (Figure 4.8):
 - a. **A1: Process Purchase Request,**
 - b. **A2: Process Order** and
 - c. **A3: Process Payment.**
2. The second level, consists of the decomposition of the aforementioned Activities in three separate sub-models, namely:
 - a. **Decomposition of Activity A1** (Figure 4.9)
 - i. **A1.1: Register Purchase Request** and
 - ii. **A1.2: Approve Purchase Request**
 - b. **Decomposition of Activity A2** (Figure 4.10)
 - i. **A2.1: Order Items,**
 - ii. **A2.2: Receive Supplier's Invoice** and
 - iii. **A2.3: Receive Ordered Items**
 - c. **Decomposition of Activity A3** (Figure 4.11)
 - i. **A3.1: Issue Payment Order,**
 - ii. **A3.2: Issue Check** and
 - iii. **A3.3: Receive Payment Receipt**

Figure 4.8 captures the first level of SYMEX model as a high-level description consisting of three Activities.

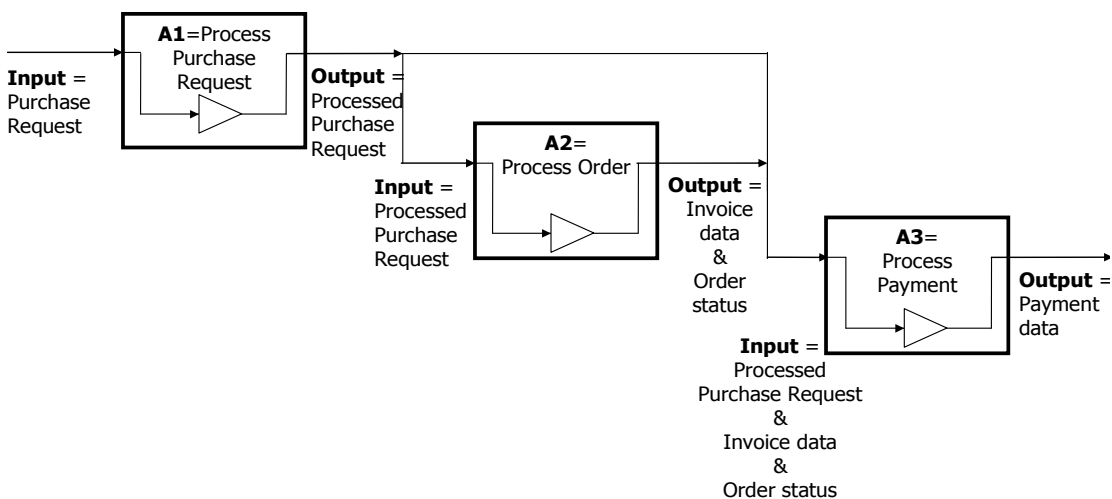


Figure 4.8: SYMEX model - High-level description of business process

4.3.2.1 Decomposition of Activity A1: Process Purchase Request

If we focus in composite **Activity A1: Process Purchase Request** and decompose it, it consists of two atomic Activities:

- (1) **A1.1: Register Purchase Request** and
- (2) **A1.2: Approve Purchase Request.**

Figure 4.9 shows the decomposed Activity A1. Note that the Input and Output of Activity A1 is used in the decomposition by Activity A1.1 and Activity A1.2 respectively. In this level we also encode actual Mechanisms of the Activities and actual Controls.

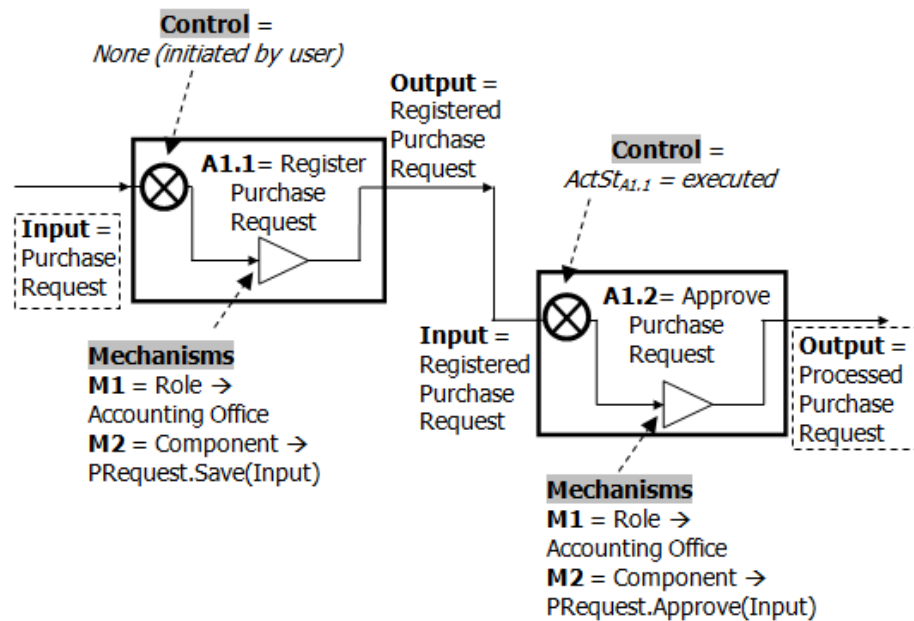


Figure 4.9: SYMEX model - Decomposition of Activity A1

4.3.2.2 Decomposition of Activity A2: Process Order

Composite **Activity A2: Process Order** is also further decomposed to atomic Activities:

- (1) **A2.1: Order Items**
- (2) **A2.2: Receive Supplier's Invoice** and
- (3) **A2.3: Receive Ordered Items**.

Figure 4.10 shows the decomposition of Activity A2.

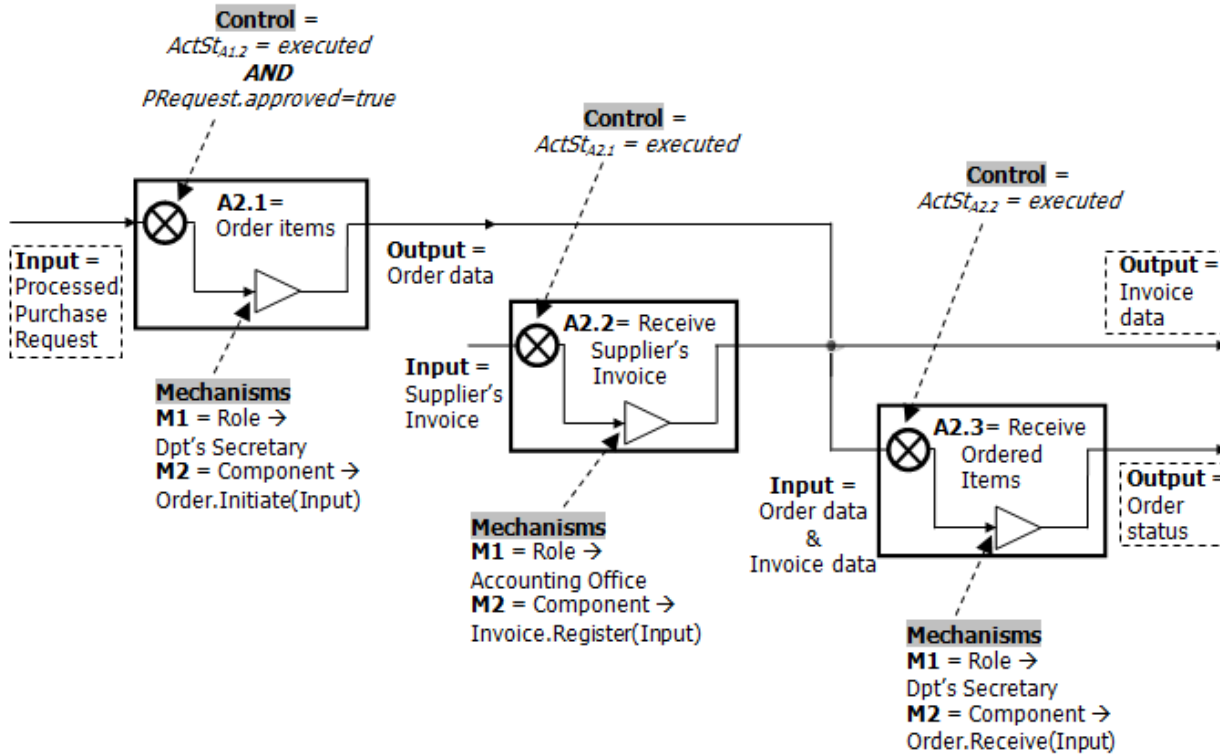


Figure 4.10: SYMEX model - Decomposition of Activity A2

4.3.2.3 Decomposition of Activity A3: Process Payment

Finally, composite **Activity A3: Process Payment** is decomposed to:

- (1) **A3.1: Issue Payment Order**,
- (2) **A3.2: Issue Check** and
- (3) **A3.3: Receive Payment Receipt**.

Figure 4.11 shows this decomposition.

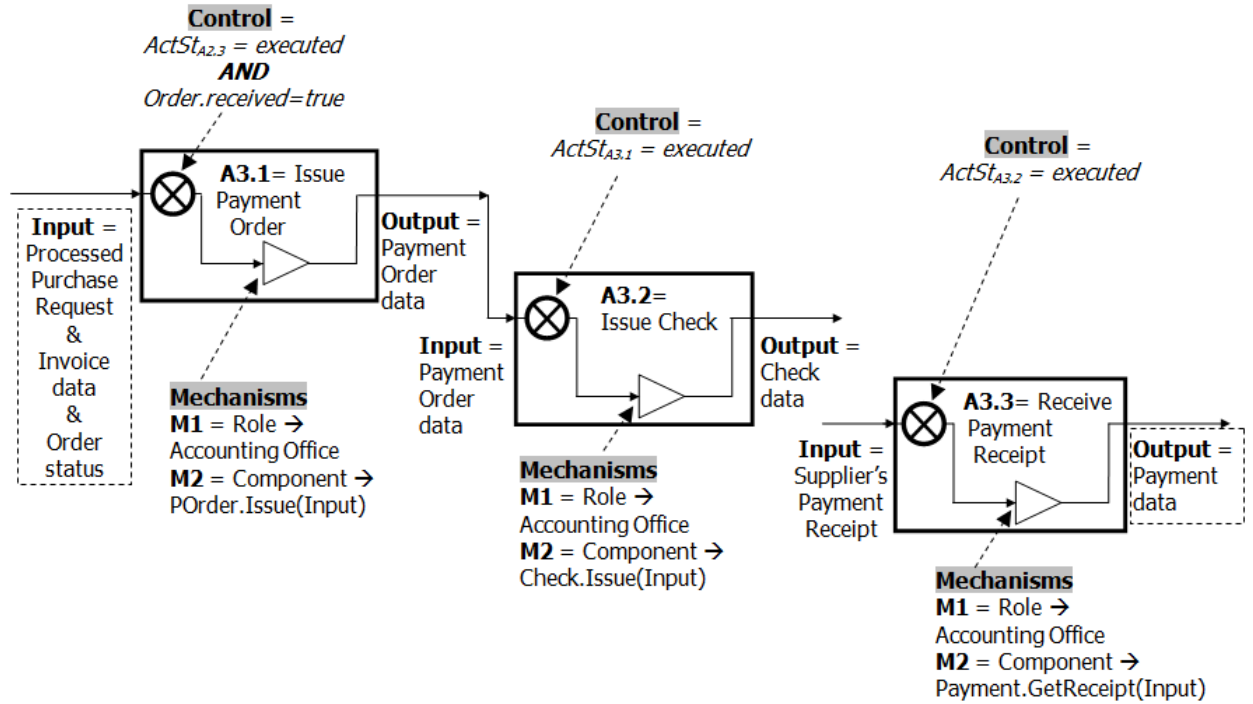


Figure 4.11: SYMEX model - Decomposition of Activity A3

4.3.3 Process description

The corresponding process description of the above SYMEX model, based on the XML schema introduced in Section 4.2.4 of this Chapter, is shown in Figure 4.12.

```

<Process ID="{0936145D091245D29DC2EE656D04DE0E112200715116}" Name="THESIS" CodeName="ExampleProcess" Description="Example of a business process">
  <EntityPool>
    <Schemas>
      <Schema ID="{A0F391B4A8A840BA919E90DB80E341DD11220071544}">
        <Name>Purchase Request</Name>
        <Description/>
        <XSD/>
      </Schema>
      <Schema ID="{F5ED1A9325554EAA8776208C75F3A4EA112200715415}">
        <Name>Processed Purchase Request</Name>
        <Description/>
        <XSD/>
      </Schema>
      <Schema ID="{02C629B897DB4B5BAF22A50E43EF6E7E112200715441}">
        <Name>Invoice data</Name>
        <Description/>
        <XSD/>
      </Schema>
      <Schema ID="{3E059881C1B34B3CAA85C5D410B3849E112200715449}">
        <Name>Order status</Name>
        <Description/>
        <XSD/>
      </Schema>
      <Schema ID="{451E3B1DF1CD46CBBEB6971FA7ECBAB3112200715515}">
        <Name>Payment data</Name>
        <Description/>
        <XSD/>
      </Schema>
      <Schema ID="{D9D6EE0FE0F34003A4C7D19B0608284B112200715624}">
        <Name>Registered Purchase Request</Name>
        <Description/>
        <XSD/>
      </Schema>
      <Schema ID="{31ADFEC59BE34E6394C495FE073F5D351122007151321}">
        <Name>Supplier's Invoice</Name>
        <Description/>
        <XSD/>
      </Schema>
      <Schema ID="{5B3A1CAF9E954CC59511151DD626FD7C1122007151559}">
        <Name>Order data</Name>
        <Description/>
        <XSD/>
      </Schema>
      <Schema ID="{18B0FFC2660E425DB5E6BAF9DD1E577A1122007152249}">
        <Name>Payment Order data</Name>
        <Description/>
        <XSD/>
      </Schema>
      <Schema ID="{C91595F791144ECA8FB17A9085B20D8D1122007152318}">
        <Name>Check data</Name>
        <Description/>
        <XSD/>
      </Schema>
      <Schema ID="{8A494A610897430FB1C4275E09FB1BD31122007152329}">
        <Name>Supplier's Payment Receipt</Name>
        <Description/>
        <XSD/>
      </Schema>
    </Schemas>
    <Controls>
      <Control ID="{264A4EA7D1E14FFC8AA2E0CB4353B33B11220071584}">
        <Name>Control of A12</Name>
        <CodeName>ControlA12</CodeName>
        <Condition>ActSt("A11")=executed</Condition>
      </Control>
      <Control ID="{AAAB64EF47374A4E898A62FA89087AC51122007151235}">
        <Name>Control of A21</Name>
        <CodeName>ControlA21</CodeName>
        <Condition>ActSt("A12")=executed AND PRequest.approved=true</Condition>
      </Control>
      <Control ID="{9B3BB7AAACE2F4DB5987AB851DFB57FBE1122007151519}">
        <Name>Control of A22</Name>
        <CodeName>ControlA22</CodeName>
        <Condition>ActSt("A21")=executed</Condition>
      </Control>
      <Control ID="{66ABBAEF58224D0CA06489E703AFE4861122007151747}">
        <Name>Control of A23</Name>
    </Controls>
  </EntityPool>
</Process>

```

```

        <CodeName>ControlA23</CodeName>
        <Condition>ActSt("A22")=executed</Condition>
    </Control>
    <Control ID="{9A935605F2184EC7ACCCB582AB930A811122007152557}">
        <Name>Control of A31</Name>
        <CodeName>ControlA31</CodeName>
        <Condition>ActSt("A23")=executed AND Order.received=true</Condition>
    </Control>
    <Control ID="{984B92A8BD274969ADEA1D835856DF2B1122007152643}">
        <Name>Control of A32</Name>
        <CodeName>ControlA32</CodeName>
        <Condition>ActSt("A31")=executed</Condition>
    </Control>
    <Control ID="{51C57080191D47EBA75DDF414738F5881122007152835}">
        <Name>Control of A33</Name>
        <CodeName>ControlA33</CodeName>
        <Condition>ActSt("A32")=executed</Condition>
    </Control>
</Controls>
<Mechanisms>
    <Mechanism ID="{58F847CA8B684AC99B4DFB62B93D93E5112200715932}">
        <Name>Component PurchaseRequest.Save</Name>
        <Description/>
        <Type>Component</Type>
        <ProgID>PurchaseRequest</ProgID>
        <Method>Save</Method>
        <Arguments>Input</Arguments>
        <Roles/>
        <XSLT/>
    </Mechanism>
    <Mechanism ID="{F34C7A96C438474E90AB571DB3DE1EBD112200715101}">
        <Name>Component PurchaseRequest.Approve</Name>
        <Description/>
        <Type>Component</Type>
        <ProgID>PurchaseRequest</ProgID>
        <Method>Approve</Method>
        <Arguments>Input</Arguments>
        <Roles/>
        <XSLT/>
    </Mechanism>
    <Mechanism ID="{78DE5A2323F245A0B7F193FCFEE41F0F112200715211}">
        <Name>Role Dept's Secretary</Name>
        <Description/>
        <Type>Role</Type>
        <ProgID/>
        <Method/>
        <Arguments/>
        <Roles>DptsSecretary</Roles>
        <XSLT/>
    </Mechanism>
    <Mechanism ID="{52948139BC234496A438FAF897085112112200715206}">
        <Name>Role Accounting Office</Name>
        <Description/>
        <Type>Role</Type>
        <ProgID/>
        <Method/>
        <Arguments/>
        <Roles>AccountingOffice</Roles>
        <XSLT/>
    </Mechanism>
    <Mechanism ID="{47CFD8CD48844F148E1E792A71E9299D1122007152413}">
        <Name>Component PaymentOrder.Issue</Name>
        <Description/>
        <Type>Component</Type>
        <ProgID>PaymentOrder</ProgID>
        <Method>Issue</Method>
        <Arguments>Input</Arguments>
        <Roles/>
        <XSLT/>
    </Mechanism>
    <Mechanism ID="{3A785AF9A4E64FFFB27E89B6D693D1901122007152431}">
        <Name>Component Check.Issue</Name>
        <Description/>
        <Type>Component</Type>
        <ProgID>Check</ProgID>
        <Method>Issue</Method>
        <Arguments>Input</Arguments>
        <Roles/>
        <XSLT/>
    </Mechanism>
    <Mechanism ID="{C4C7FE4F75354055BF55C23F6BE7B91F1122007152445}">
        <Name>Component Payment.GetReceipt</Name>
        <Description/>
        <Type>Component</Type>
        <ProgID>Payment</ProgID>
        <Method>GetReceipt</Method>
    </Mechanism>

```

```

        <Arguments>Input</Arguments>
        <Roles/>
        <XSLT/>
    </Mechanism>
    <Mechanism ID="{C4C7FE4F753540KJGHKSFHSD981F1122007152445}">
        <Name>Component Order.Initiate</Name>
        <Description/>
        <Type>Component</Type>
        <ProgID>Order</ProgID>
        <Method>Initiate</Method>
        <Arguments>Input</Arguments>
        <Roles/>
        <XSLT/>
    </Mechanism>
    <Mechanism ID="{C4C7FE4567890LKSDFHJKLSDF98341122007152445}">
        <Name>Component Invoice.Register</Name>
        <Description/>
        <Type>Component</Type>
        <ProgID>Invoice</ProgID>
        <Method>Register</Method>
        <Arguments>Input</Arguments>
        <Roles/>
        <XSLT/>
    </Mechanism>
    <Mechanism ID="{KDFSGSDIOFOF097ADSGF354T4GF41122007152445}">
        <Name>Component Order.Receive</Name>
        <Description/>
        <Type>Component</Type>
        <ProgID>Order</ProgID>
        <Method>Receive</Method>
        <Arguments>Input</Arguments>
        <Roles/>
        <XSLT/>
    </Mechanism>
</Mechanisms>
</EntityPool>
<Activities>
    <Activity ID="{740E9B3BE07F47C6A92E6BAFFD06A489112200715335}" Name="Process Purchase Request" Code-
Name="" Description="" PendingJobInfo="" ControlSynch="">
        <Inputs>
            <Item ID="{A0F391B4A8A840BA919E90DB80E341DD11220071544}" />
        </Inputs>
        <Outputs RuntimeMainOutput="{F5ED1A9325554EAA8776208C75F3A4EA112200715415}">
            <Item ID="{F5ED1A9325554EAA8776208C75F3A4EA112200715415}" />
        </Outputs>
        <Controls/>
        <Mechanisms>
            <Item ID="{52948139BC234496A438FAF897085112112200715206}" />
        </Mechanisms>
    </Activity>
    <Activity ID="{55C7252CB77849DC95F9484B00A48C34112200715345}" Name="Process Order" CodeName=""
Description="" PendingJobInfo="" ControlSynch="">
        <Inputs>
            <Item ID="{F5ED1A9325554EAA8776208C75F3A4EA112200715415}" />
        </Inputs>
        <Outputs RuntimeMainOutput="{02C629B897DB4B5BAF22A50E43EF6E7E112200715441}">
            <Item ID="{02C629B897DB4B5BAF22A50E43EF6E7E112200715441}" />
            <Item ID="{3E059881C1B34B3CAA85C5D410B3849E112200715449}" />
        </Outputs>
        <Controls/>
        <Mechanisms>
            <Item ID="{52948139BC234496A438FAF897085112112200715206}" />
            <Item ID="{78DE5A2323F245A0B7F193FCFEE41F0F112200715211}" />
        </Mechanisms>
    </Activity>
    <Activity ID="{F5E7A250DDA142EA8081D94AE1367BE0112200715354}" Name="Process Payment" CodeName=""
Description="" PendingJobInfo="" ControlSynch="">
        <Inputs>
            <Item ID="{F5ED1A9325554EAA8776208C75F3A4EA112200715415}" />
            <Item ID="{02C629B897DB4B5BAF22A50E43EF6E7E112200715441}" />
            <Item ID="{3E059881C1B34B3CAA85C5D410B3849E112200715449}" />
        </Inputs>
        <Outputs RuntimeMainOutput="{451E3B1DF1CD46CBBEB6971FA7ECBAB3112200715515}">
            <Item ID="{451E3B1DF1CD46CBBEB6971FA7ECBAB3112200715515}" />
        </Outputs>
        <Controls/>
        <Mechanisms>
            <Item ID="{52948139BC234496A438FAF897085112112200715206}" />
        </Mechanisms>
    </Activity>
    <Activity ID="{F7A00E39831E4732833AED9C543BC066112200715556}" Name="Register Purchase Request" Code-
Name="RegisterPurchaseRequest" Description="" PendingJobInfo="" ControlSynch="">
        <Inputs>
            <Item ID="{A0F391B4A8A840BA919E90DB80E341DD11220071544}" />
        </Inputs>
        <Outputs RuntimeMainOutput="{D9D6EE0FE0F34003A4C7D19B0608284B112200715624}">

```

```

        <Item ID="{D9D6EE0FE0F34003A4C7D19B0608284B112200715624}"/>
    </Outputs>
    <Controls/>
    <Mechanisms>
        <Item ID="{58F847CA8B684AC99B4DFB62B93D93E5112200715932}"/>
        <Item ID="{52948139BC234496A438FAF897085112112200715206}"/>
    </Mechanisms>
</Activity>
<Activity ID="{8981E2D2CB99483C8837A5B0C277DA3111220071566}" Name="Approve Purchase Request" Code-
Name="ApprovePurchaseRequest" Description="" PendingJobInfo="" ControlSynch="ControlA12">
    <Inputs>
        <Item ID="{D9D6EE0FE0F34003A4C7D19B0608284B112200715624}"/>
    </Inputs>
    <Outputs RuntimeMainOutput="{F5ED1A9325554EAA8776208C75F3A4EA112200715415}">
        <Item ID="{F5ED1A9325554EAA8776208C75F3A4EA112200715415}"/>
    </Outputs>
    <Controls>
        <Item ID="{264A4EA7D1E14FFC8AA2E0CB4353B33B11220071584}"/>
    </Controls>
    <Mechanisms>
        <Item ID="{F34C7A96C438474E90AB571DB3DE1EBD112200715101}"/>
        <Item ID="{52948139BC234496A438FAF897085112112200715206}"/>
    </Mechanisms>
</Activity>
<Activity ID="{18780860411E43008E93A3B26DEC12F5112200715111}" Name="Order Items" Code-
Name="OrderItems" Description="" PendingJobInfo="" ControlSynch="ControlA21">
    <Inputs>
        <Item ID="{F5ED1A9325554EAA8776208C75F3A4EA112200715415}"/>
    </Inputs>
    <Outputs RuntimeMainOutput="{5B3A1CAF9E954CC59511151DD626FD7C1122007151559}">
        <Item ID="{5B3A1CAF9E954CC59511151DD626FD7C1122007151559}"/>
    </Outputs>
    <Controls>
        <Item ID="{AAAB64EF47374A4E898A62FA89087AC51122007151235}"/>
    </Controls>
    <Mechanisms>
        <Item ID="{78DE5A2323F245A0B7F193FCFEE41F0F112200715211}"/>
        <Item ID="{C4C7FE4F753540KJGHKSFHSD981F1122007152445}"/>
    </Mechanisms>
</Activity>
<Activity ID="{3C4CF23A64234C8E83ED4DE0905048421122007151311}" Name="Receive Supplier's Invoice"
CodeName="ReceiveSuppliersInvocie" Description="" PendingJobInfo="" ControlSynch="ControlA22">
    <Inputs>
        <Item ID="{31ADFE59BE34E6394C495FE073F5D351122007151321}"/>
    </Inputs>
    <Outputs RuntimeMainOutput="{02C629B897DB4B5BAF22A50E43EF6E7E112200715441}">
        <Item ID="{02C629B897DB4B5BAF22A50E43EF6E7E112200715441}"/>
    </Outputs>
    <Controls>
        <Item ID="{9B3BB7AACE2F4DB5987AB851DFB57FBE1122007151519}"/>
    </Controls>
    <Mechanisms>
        <Item ID="{52948139BC234496A438FAF897085112112200715206}"/>
        <Item ID="{C4C7FE4567890LKSDFHJKLSDFJF98341122007152445}"/>
    </Mechanisms>
</Activity>
<Activity ID="{37822B583261498EAC05113898A968B31122007151543}" Name="Receive Ordered Items" Code-
Name="ReceiveOrderedItems" Description="" PendingJobInfo="" ControlSynch="ControlA23">
    <Inputs>
        <Item ID="{5B3A1CAF9E954CC59511151DD626FD7C1122007151559}"/>
        <Item ID="{02C629B897DB4B5BAF22A50E43EF6E7E112200715441}"/>
    </Inputs>
    <Outputs RuntimeMainOutput="{3E059881C1B34B3CAA85C5D410B3849E112200715449}">
        <Item ID="{3E059881C1B34B3CAA85C5D410B3849E112200715449}"/>
    </Outputs>
    <Controls>
        <Item ID="{66ABBAEF58224D0CA06489E703AFE4861122007151747}"/>
    </Controls>
    <Mechanisms>
        <Item ID="{78DE5A2323F245A0B7F193FCFEE41F0F112200715211}"/>
        <Item ID="{KDFSGSDIOF0F097ADSGF354T4GF41122007152445}"/>
    </Mechanisms>
</Activity>
<Activity ID="{D2450EF2B9E84D38B304400E711B542C1122007152158}" Name="Issue Payment Order" Code-
Name="IssuePaymentOrder" Description="" PendingJobInfo="" ControlSynch="ControlA31">
    <Inputs>
        <Item ID="{F5ED1A9325554EAA8776208C75F3A4EA112200715415}"/>
        <Item ID="{02C629B897DB4B5BAF22A50E43EF6E7E112200715441}"/>
        <Item ID="{3E059881C1B34B3CAA85C5D410B3849E112200715449}"/>
    </Inputs>
    <Outputs RuntimeMainOutput="{18B0FFC2660E425DB5E6BAF9DD1E577A1122007152249}">
        <Item ID="{18B0FFC2660E425DB5E6BAF9DD1E577A1122007152249}"/>
    </Outputs>
    <Controls>
        <Item ID="{9A935605F2184EC7ACCCB582AB930A811122007152557}"/>
    </Controls>

```

```

        <Mechanisms>
            <Item ID="{52948139BC234496A438FAF897085112112200715206}" />
            <Item ID="{47CFD8CD48844F148E1E792A71E9299D1122007152413}" />
        </Mechanisms>
    </Activity>
    <Activity ID="{F8D0BCDE1D724EA38508EBDB7C91D245112200715224}" Name="Issue Check" Code-
Name="IssueCheck" Description="" PendingJobInfo="" ControlSynch="ControlA32">
        <Inputs>
            <Item ID="{18B0FFC2660E425DB5E6BAF9DD1E577A1122007152249}" />
        </Inputs>
        <Outputs RuntimeMainOutput="{C91595F791144ECA8FB17A9085B20D8D1122007152318}">
            <Item ID="{C91595F791144ECA8FB17A9085B20D8D1122007152318}" />
        </Outputs>
        <Controls>
            <Item ID="{984B92A8BD274969ADEA1D835856DF2B1122007152643}" />
        </Controls>
        <Mechanisms>
            <Item ID="{52948139BC234496A438FAF897085112112200715206}" />
            <Item ID="{3A785AF9A4E64FFFB27E89B6D693D1901122007152431}" />
        </Mechanisms>
    </Activity>
    <Activity ID="{52BC4F5FCB084618A60C062CF7FEB04B1122007152212}" Name="Receive Payment Receipt"
CodeName="ReceivePaymentReceipt" Description="" PendingJobInfo="" ControlSynch="ControlA33">
        <Inputs>
            <Item ID="{8A494A610897430FB1C4275E09FB1BD31122007152329}" />
        </Inputs>
        <Outputs RuntimeMainOutput="{451E3B1DF1CD46CBEB6971FA7ECBAB3112200715515}">
            <Item ID="{451E3B1DF1CD46CBEB6971FA7ECBAB3112200715515}" />
        </Outputs>
        <Controls>
            <Item ID="{51C57080191D47EBA75DDF414738F5881122007152835}" />
        </Controls>
        <Mechanisms>
            <Item ID="{52948139BC234496A438FAF897085112112200715206}" />
            <Item ID="{C4C7FE4F75354055BF55C23F6BE7B91F1122007152445}" />
        </Mechanisms>
    </Activity>
</Activities>
</Process>

```

Figure 4.12: Process description in XML format

4.3.4 Discussion

As shown in Figure 4.8, the first level of SYMEX model does not expose any implementation details (Controls or Mechanisms of Activities). This level represents the business process in an abstract layer, showing the main classification of Activities and the information flow between them. The three Activities are then further decomposed to three sub-models as shown in Figure 4.9, Figure 4.10, and Figure 4.11. The models at this decomposed level may seem as separate but they are ultimately connected through the relationships between their parent Activities. Moreover, each of these models explicitly encodes how its parent Activity is implemented, modelling the Controls and Mechanisms of the process. At the end, it is the information of this second level that encodes the execution semantics of the process, which will be used for execution purposes by the workflow inference engine.

The example of modelling a real business process derived from the accounting domain, was used in order to demonstrate the basic methodology and principles of SYMEX modelling. For this, we modelled only a small part of the University's business process, in order to avoid many levels of decomposition and an excessive set of models that would prevent the reader from easily following and understanding SYMEX modelling. However, in more extended business processes the levels of decomposition would be much more. If for example we would like to create a SYMEX model covering all activities inside a University, we would have a high-level description with Activities: **A1: Finance Office, A2: Registry Office, A3: Departments** etc. These Activities would be further decomposed to **A1.1 Tuition Fees, A1.2 Salaries, and A1.3 Payment** and so on. Therefore, it is evident that larger business processes result to more levels of decompositions and an increasing set of models. Higher-levels of decomposition always depict high-level descriptions of business processes, while final levels of decomposition always encode the execution semantics of processes.

4.4 Comparative analysis of change management

This Section presents a comparative analysis of change management between SYMEX models and UML 2 Activity Diagrams [33]. Specifically, this analysis validates that constraint-based modelling adopted by SYMEX is more efficient in terms of adaptation in comparison with modelling using linear sequences of activities. The reason for selecting UML 2 Activity Diagram for this comparative analysis is that all process modelling languages we reviewed in Chapter 2 follow the same approach, sequencing activities of processes with the construction of linear relationships by using controls flows in the models. Control flows encode traditional programming constructs such as if-the-else, cases, while loops etc. Thus UML 2 Activity Diagram is representative of commonly used modelling languages in terms of change management of process models.

4.4.1 Travel agency scenario

For the analysis, we employ a travel agency scenario [177] as follows. A travel agent undertakes to fulfil the requirements of a trip for a customer, by arranging both the customer's flight and hotel accommodation. The customer requires that the flight's actual date must be as close as possible to his desired date for travel, and that there must be available hotel accommodation for that date. The task of the travel agent is to orchestrate external services for flight and hotel booking in a way that meets the customer requirements. There are mutual dependencies between the services, as hotel reservation can only be made after a suitable flight has been found. However, the flight service depends also on the hotel booking service, as if there is no hotels' availability for the found date then an alternative flight must be sought.

The adaptation that will take place is a case where the travel agency extends its range of services by providing additionally a car booking service. The newly added service of car booking, adds a new set of dependencies to the existing services. A car booking can only be made after a hotel reservation has been made. However, based on the customer's demand if there is no cars' availability then alternative flights and hotels must be sought.

4.4.2 Modelling with UML 2 Activity Diagram

4.4.2.1 Initial UML 2 Activity Diagram

In this Section we model the travel agency scenario using UML 2 Activity diagram, as shown in Figure 4.13.

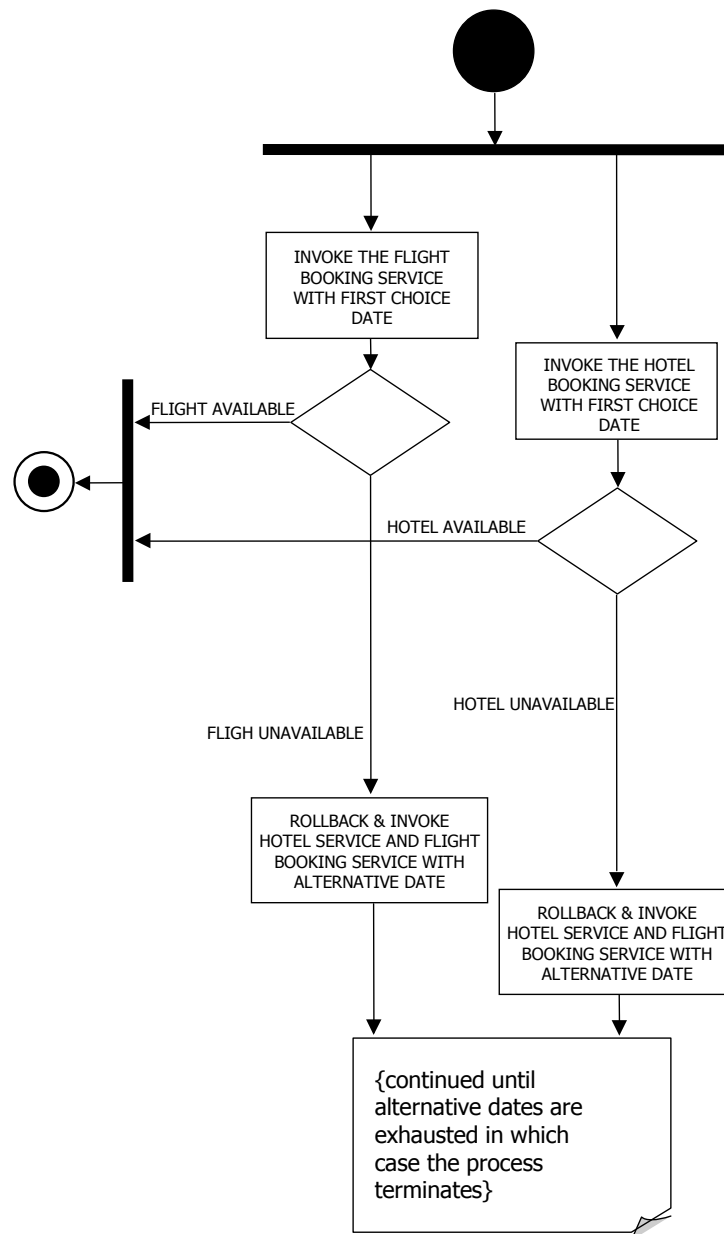


Figure 4.13: UML 2 Activity Diagram for travel agency scenario [177]

The UML 2 Activity Diagram for the travel agency process, explicitly models that when the process starts, two parallel activities are initiated; “Invoke flight booking service” and “Invoke hotel booking service”. When acceptable date is found for both flight and hotel accommodation, the process ends. Otherwise, the process roll backs and invokes again the two activities till an acceptable date is reached.

4.4.2.2 Adaptation of UML 2 Activity Diagram

In order to introduce the new service in the previous UML 2 Activity Diagram we are required to do modifications in several places as now three conditions need to be satisfied, as shown in Figure 4.14.

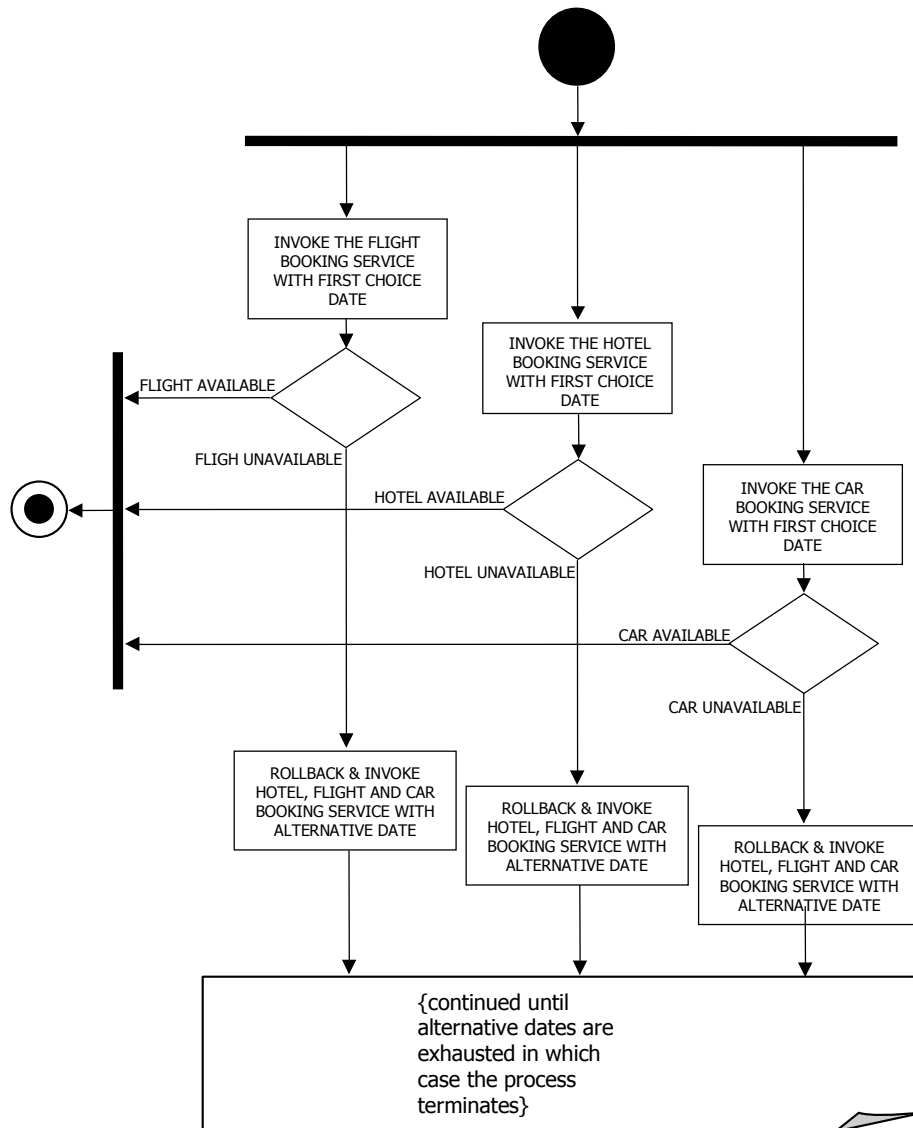


Figure 4.14: Adapted UML 2 Activity Diagram for travel agency scenario [177]

In the adapted UML 2 Activity Diagram, the newly added activity “Invoke car booking service” is explicitly added as a parallel activity to “Invoke flight booking service” and “Invoke hotel booking service”. Moreover, the termination of the process now depends on the synchronization of outputs from all the three activities and in case that one of them returns “no availability” or an unacceptable date, the process roll backs and invokes again the three activities till an acceptable date is reached.

4.4.3 SYMEX modelling

4.4.3.1 Initial SYMEX model

In this Section, we model the travel agency scenario using SYMEX, as shown in Figure 4.15.

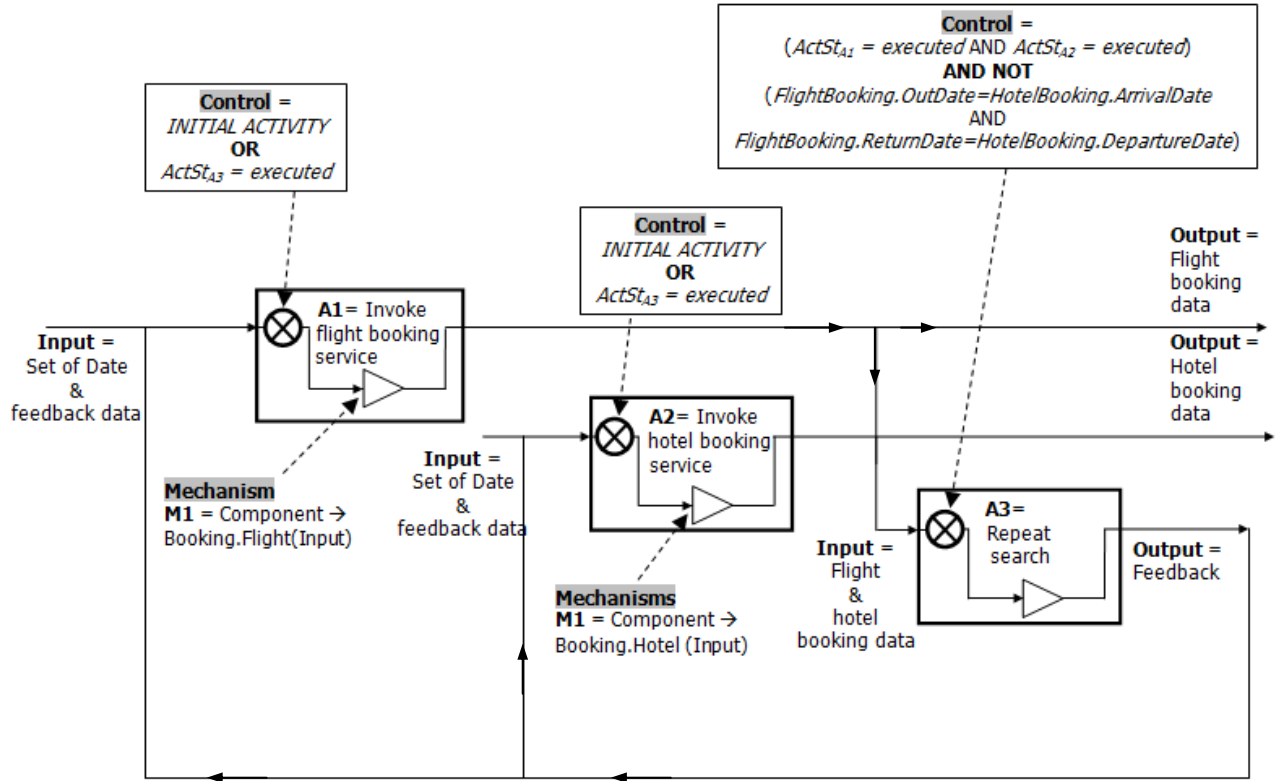


Figure 4.15: SYMEX model for travel agency scenario

The SYMEX model for the travel agency process includes three Atomic Activities, namely **A1: Invoke flight booking service**, **A2: Invoke hotel booking service** and **A3: Repeat search**. Activity A3 is activated when its Control evaluates to 'true'. The constraints specified by the Control are as follows:

$$\begin{aligned}
 & (ActSt_{A1} = executed \text{ AND } ActSt_{A2} = executed) \\
 & \text{AND NOT} \\
 & (FlightBooking.OutDate=HotelBooking.ArrivalDate \\
 & \text{AND} \\
 & FlightBooking.ReturnDate=HotelBooking.DepartureDate)
 \end{aligned}$$

Formula 4.1: Control of Activity A3 at the initial SYMEX model

The Control of Activity A3 encodes the business rule that was described in the travel agency scenario: travel agent can only confirm a booking after a suitable set of dates has been found for both flight and hotel booking. Otherwise, an alternative set of dates must be sought.

4.4.3.2 Adaptation of SYMEX model

In order to introduce the new service in the previous SYMEX model, we create the model shown in Figure 4.16.

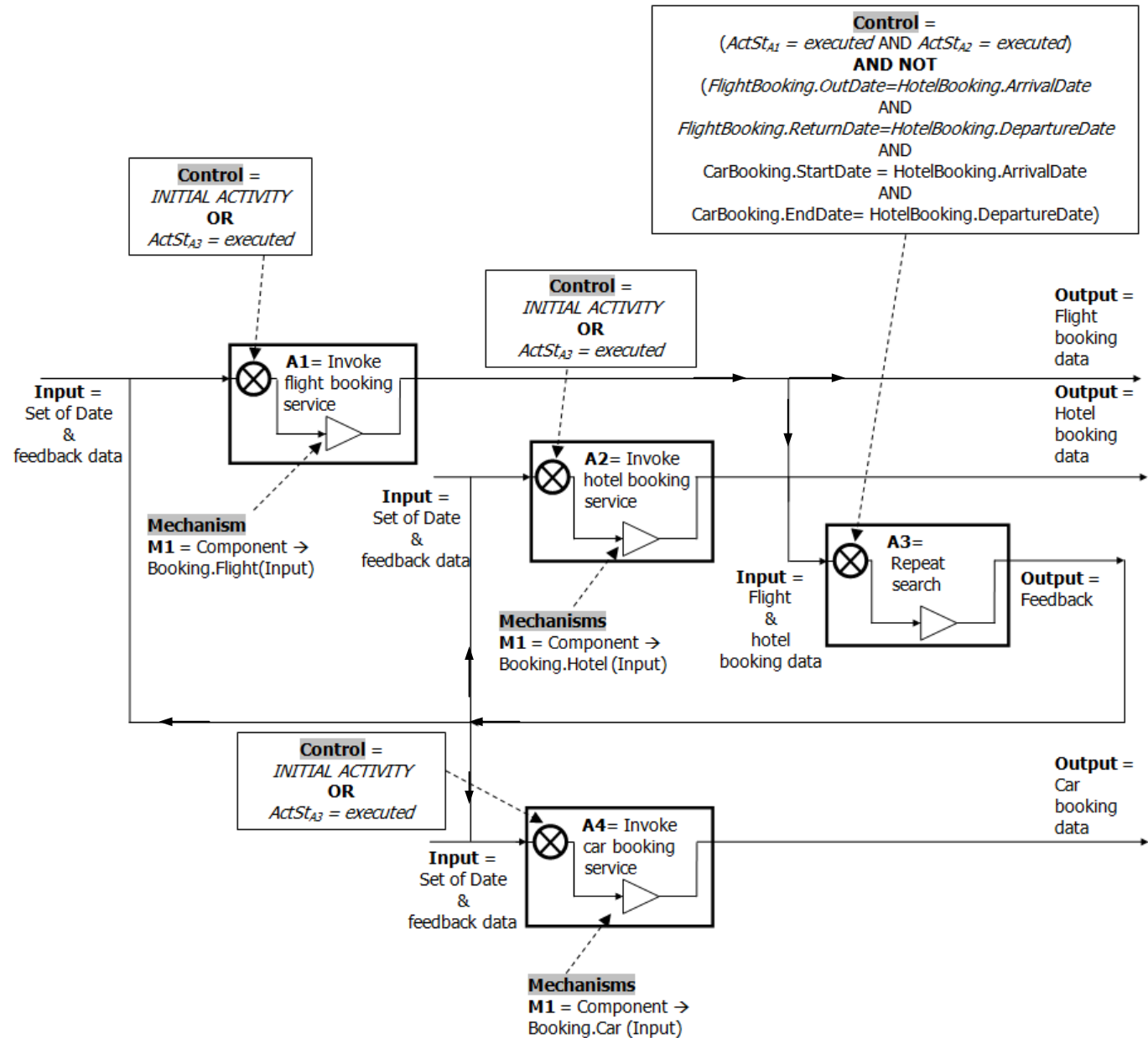


Figure 4.16: Adapted SYMEX model for travel agency scenario

In the adapted SYMEX model, the newly added service "Invoke car booking service" is added as a new Activity A4, with its Input and Output. Activities A1 and A2 remain the same while the Control of Activity **A3: Repeat search** is updated accordingly as follows:

$(ActSt_{A1} = executed \text{ AND } ActSt_{A2} = executed)$
AND NOT
 $(FlightBooking.OutDate = HotelBooking.ArrivalDate$
AND
 $FlightBooking.ReturnDate = HotelBooking.DepartureDate$
AND
 $CarBooking.StartDate = HotelBooking.ArrivalDate$
AND
 $CarBooking.EndDate = HotelBooking.DepartureDate)$

Formula 4.2: Control of Activity A3 at the adapted SYMEX model

4.4.4 Discussion of results

In Section 4.4.2 we modelled the travel agency scenario using UML 2 Activity Diagram technique. In the adaptation of the scenario, we needed to add one more activity. The result was a total number of five (5) changes to the initial model, as shown in Figure 4.17:

1. Redefinition of the concurrent activities after the initial activity
2. Redefinition of the concurrent activities before the Final Activity
3. Addition of a new Activity with the decision based on the car availability
4. Update of Activity that occurs when there is no flight availability
5. Update of Activity that occurs when there is no hotel availability

On the other hand, in Section 4.4.3, we modelled the same travel agency scenario using SYMEX modelling and in the adaptation, we had a total number of two (2) changes, as shown in Figure 4.18:

1. Addition of new Activity A4=Invoke car booking service, defining its Input, Output, Control and execution Mechanism
2. Update of Control of Activity A3

The conclusion that can be made is that modelling a business process with the construction of linear sequences of activities, like in the UML 2 Activity Diagrams, creates problems in change management of processes. When a business process is sequenced using traditional programming constructs (if-then-else, cases, while-loops etc.), the process designer needs to resolve all the sequence of activities and the dependencies between them in order to adapt the process to the new business needs. In large and more complex business processes, the adaptation process becomes a challenge and creates high maintenance costs.

On the other hand, constraint-based reasoning of SYMEX models proves to be easier and quicker to adapt. In particular, SYMEX models capture Activities and their information flow and do not provide any control flow constructs. The business rules are encoded in terms of constraints at the Control of each Activity. In this way, Activities are not explicitly sequenced. As we showed in the example of the travel agency scenario, the absence of control flow constructs eliminates the de-

dependencies between Activities, enabling easier adaptation of the process by just adding the new Activity and setting the relevant constraints at the Controls. Process designers working with SYMEX models need to focus only on the change that needs to be done, not having to interfere with any other irrelevant Activities. Thus, constraint-based approach adopted by SYMEX modelling enables easy and quick adaptations of process models in response to business change.

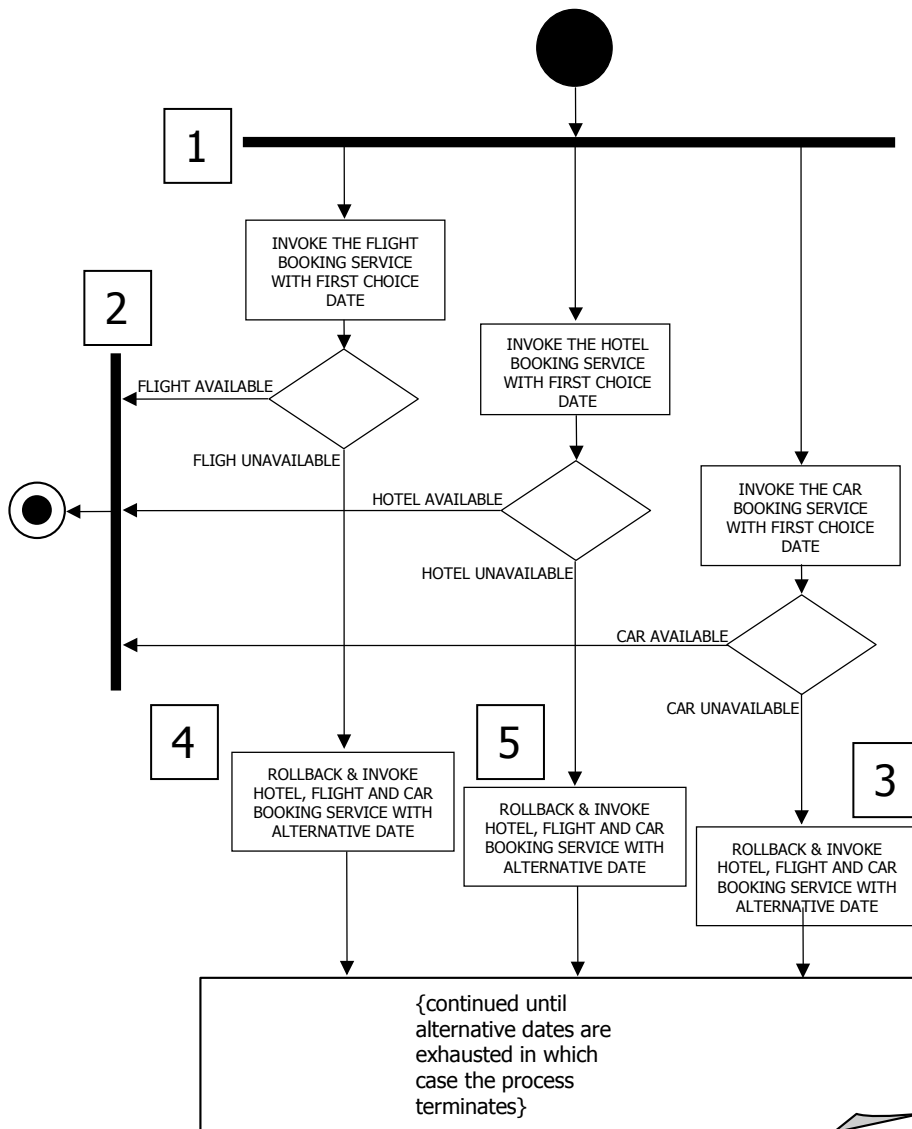


Figure 4.17: Changes to UML 2 Activity Diagram for the travel agency scenario

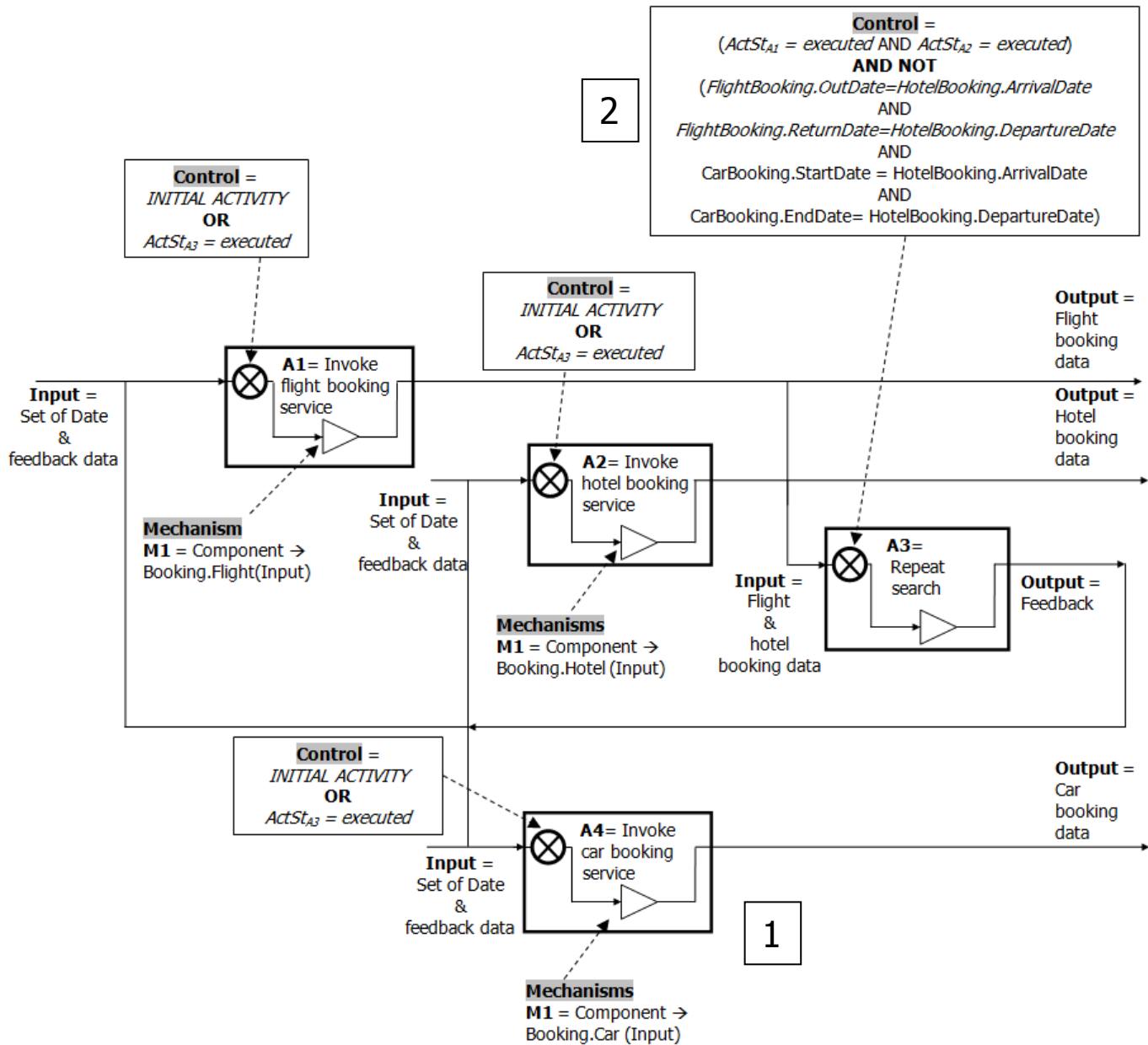


Figure 4.18: Changes to SYMEX model for the travel agency scenario

4.5 Workflow inference engine

This Section describes the theory and the algorithm of our implemented workflow inference engine, which we use for the execution of SYMEX models.

4.5.1 Theory

As we discussed in Section 2.4.2 of Chapter 2, constraint-based reasoning is a problem-solving approach based on deductive reasoning. In our research, we conceptualize the problem of workflow execution in terms of a constraint satisfaction problem [162]. For the definition of the engine's algorithm, we focus on the execution semantics defined in Chapter 3, and briefly presented below:

- An atomic Activity can be only in one of three possible states '**available**' or '**unavailable**' or '**executed**':

$$\begin{aligned} \text{ActSt}_{A_i} = f: A_i \rightarrow \{\text{available} \vee \text{unavailable} \vee \text{executed}\}, A_i \in \text{AtAct}(\Sigma) \\ \text{and} \\ \text{ActSt}(\Sigma): \forall i, 1 \leq i \leq n, A_i \in \Sigma, \text{ActSt}_{A_i} \in \text{ActSt}(\Sigma), A_i \in \text{AtAct}(\Sigma) \end{aligned}$$

Formula 4.3: Definition of Activity States for a business process Σ

- Each atomic Activity of a SYMEX model can be executed whenever it is '**available**':

$$\begin{aligned} (\forall C_i \in \mathbf{C}(A_i), \text{eval}(C_i) = \text{true}), 1 \leq i \leq n, A_i \in \text{AtAct}(\Sigma) \\ \Rightarrow \text{ActSt}_{A_i} = f: A_i \rightarrow \{\text{available}\} \end{aligned}$$

Formula 4.4: Definition of status 'available**' for Activity A_i**

- Inputs are used by the execution Mechanisms of atomic Activities in order to produce their Output:

$$\begin{aligned} \forall A_i \in \Sigma, A_i \in \text{AtAct}(\Sigma) : (\exists j \in \mathbf{I}(A_i) \wedge k \in \mathbf{O}(A_i) \wedge m \in \mathbf{M}(A_i)) \\ \Rightarrow m = f: j \rightarrow k \end{aligned}$$

Formula 4.5: Definition of execution Mechanism of Activity A_i

- The Inputs and the set of Activity States are used by Controls of atomic Activities to constrain their execution:

$$\begin{aligned} \forall A_i \in \Sigma, A_i \in \text{AtAct}(\Sigma) : (\exists j \in \mathbf{I}(A_i) \wedge k \in \mathbf{C}(A_i)) \\ \Rightarrow k = f: (j \vee \text{ActSt}(\Sigma)) \rightarrow \{\text{True} \vee \text{False}\} \end{aligned}$$

Formula 4.6: Definition of Control of Activity A_i

4.5.2 Constraint-based algorithm

The main difference between common process execution engines, such as BPEL4WS engines [183-185] and our inference engine is that the execution of our engine is based on constraint-satisfaction and not on common programming structures. At each step of the execution, the engine checks all the candidate activities to be executed, i.e. all the Activities of the Process except from the 'executed' ones, and eventually executes those Activities that their constraints are being

satisfied. The sequence of execution is thus implicit and can only be recorded during execution and not in advance.

This Section describes the constraint-based algorithm of the workflow inference engine, using the formulation defined in Chapter 3. We emphasize on the logic, so common programming error checking of the actual source code are not included in the following pseudo-code listing.

```

Public Sub Execute (ActCodename, oInput, oProcess)
    '-- Find the Activity  $A_i \in \Sigma$ , based on the Codename of the Activity
     $f: ( \text{ActCodename} ) \rightarrow A_i \in \Sigma$ 
    '-- Check if the state of Activity  $A_i \in \Sigma$ , is available and thus can be executed
    if  $\text{ActSt}_{A_i} \neq f: ( A_i ) \rightarrow \text{available}$  then Exit Sub

    '-- For each Mechanism of Activity, check its type and act accordingly
     $\forall i, 1 \leq i \leq n, M_i \in M(A)$ 
        select case type of  $M_i$ 
            case Component
                call  $M_i.\text{component.method}(\text{arguments})$ 
                 $O_i \in O(A), O_i = \text{return of execution}$ 
            case XSLT
                apply XSLT to  $I_i \in I(A)$ 
                 $O_i \in O(A), O_i = \text{result of transformation}$ 
        end select

    '-- Set the state of Activity  $A_i \in \Sigma$ , to executed
     $\text{ActSt}_{A_i} = f: ( A_i ) \rightarrow \text{executed}$ 
    '-- Check which other Activities of the process are available
    Call CheckAvailableActivities (oProcess)
    '--Create and Return XML
    Return XML with status, error details
End Sub

Private Sub CheckAvailableActivities (oProces)
    '-- Check each Activity of the Process whether it is 'available'
     $\forall i, 1 \leq i \leq n, A_i \in \Sigma$ 
        '-- If all Inputs are 'available', its Controls evaluate to 'True' and Activity has
        not been executed, it is considered as 'available' and is being executed
        if  $(\forall I_i \in I(A_i), \text{available}(I_i) = \text{true}) \wedge$ 
             $(\forall C_i \in C(A_i), \text{eval}(C_i) = \text{true}) \wedge$ 
             $\text{ActSt}_{A_i} \neq f: ( A_i ) \rightarrow \text{executed}$  then
             $f: ( A_i ) \rightarrow \text{available}$ 
            '-- Call recursively method Execute
            Call Execute ( $A_i.\text{Codename}, I_i \in I(A_i), \text{oProcess}$ )
        end if
    End Sub

```

Table 4.1: Pseudo-code listing of constraint-based algorithm

4.5.3 Implementation

After describing the constraint-based algorithm of the inference engine, this Section describes the design and implementation of the engine. Current implementation is in MS Visual Basic 6, but the code could be easily immigrated to Java [186] or .Net platform, as there is no use of Visual Basic 6 specific controls. The following UML class diagrams of the implemented workflow inference engine, illustrate their attributes, operations and relationships.

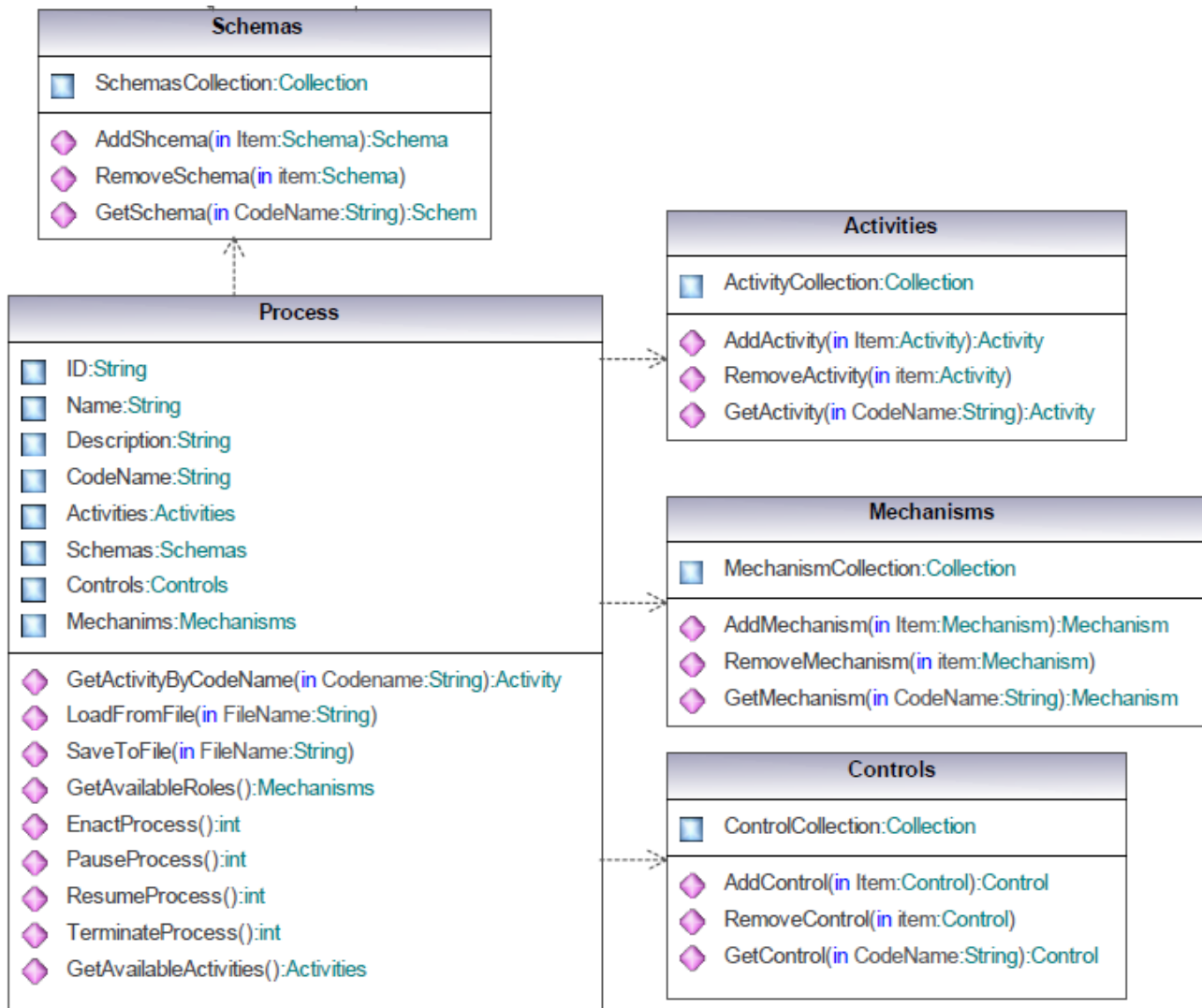


Figure 4.19: UML Class Diagram of the implemented workflow inference engine

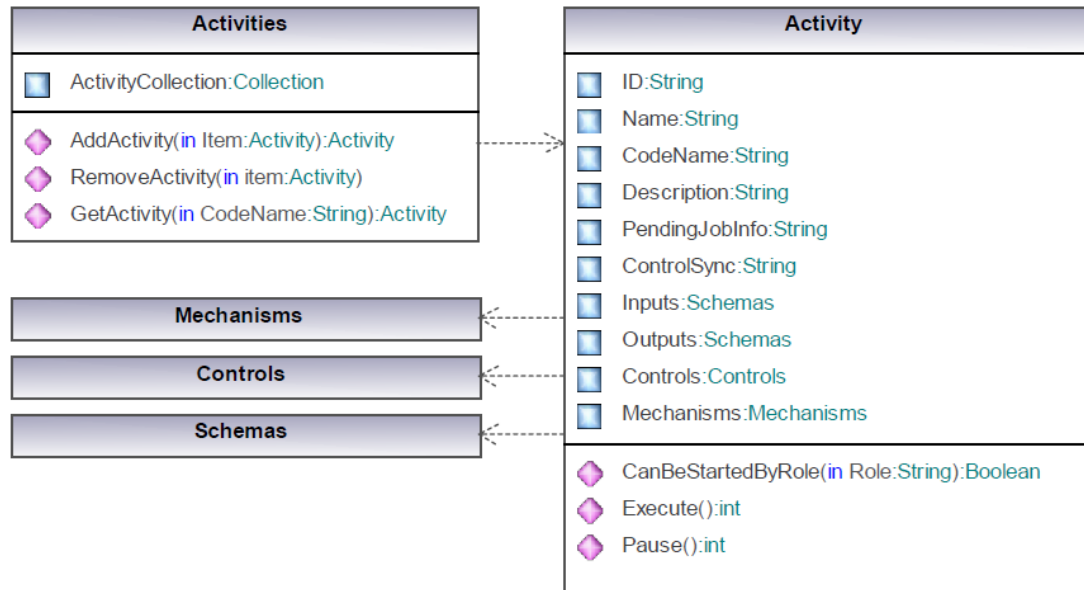


Figure 4.20: UML Class Diagram: Activities and Activity classes

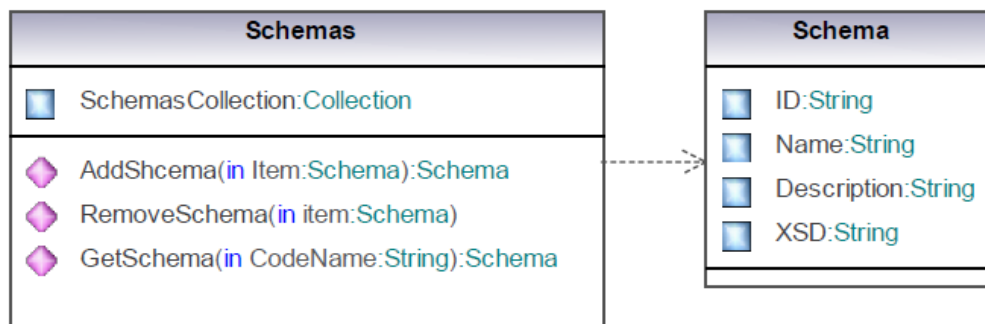


Figure 4.21: UML Class Diagram: Schemas and Schema classes

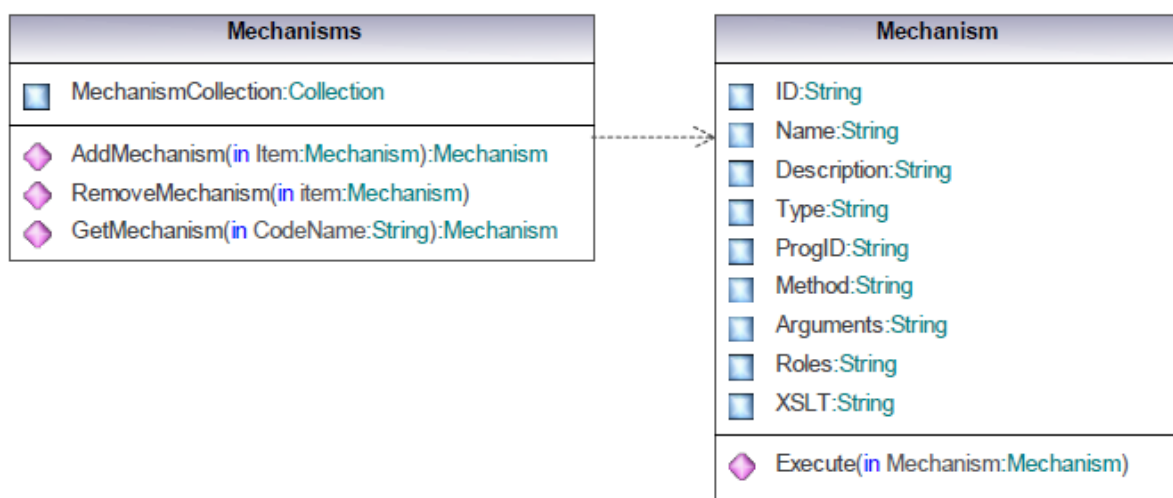


Figure 4.22: UML Class Diagram: Mechanisms and Mechanism classes

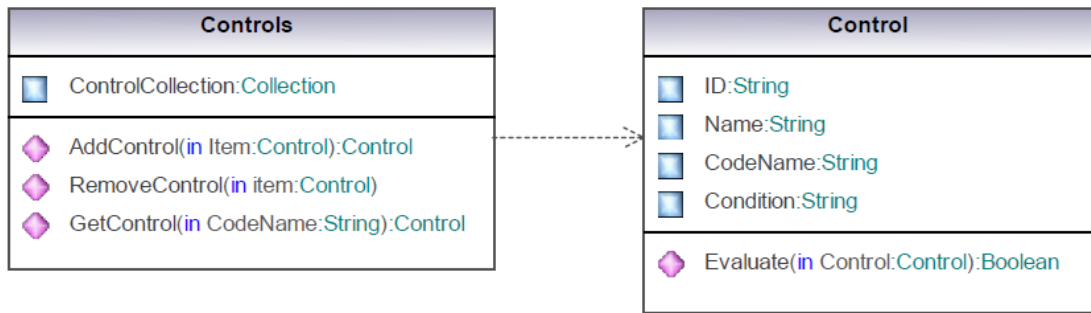


Figure 4.23: UML Class Diagram: Controls and Control classes

As far as the data persistence of the inference engine is concerned, we use the Microsoft's SQL Server 2000 as our Relational Database Management System (RDBMS). Again, we use pure SQL structures and constructs and we could easily use any other available RDBMS, such as Oracle, MySQL etc. The following figure shows the tables in the database, their fields, and relationships.

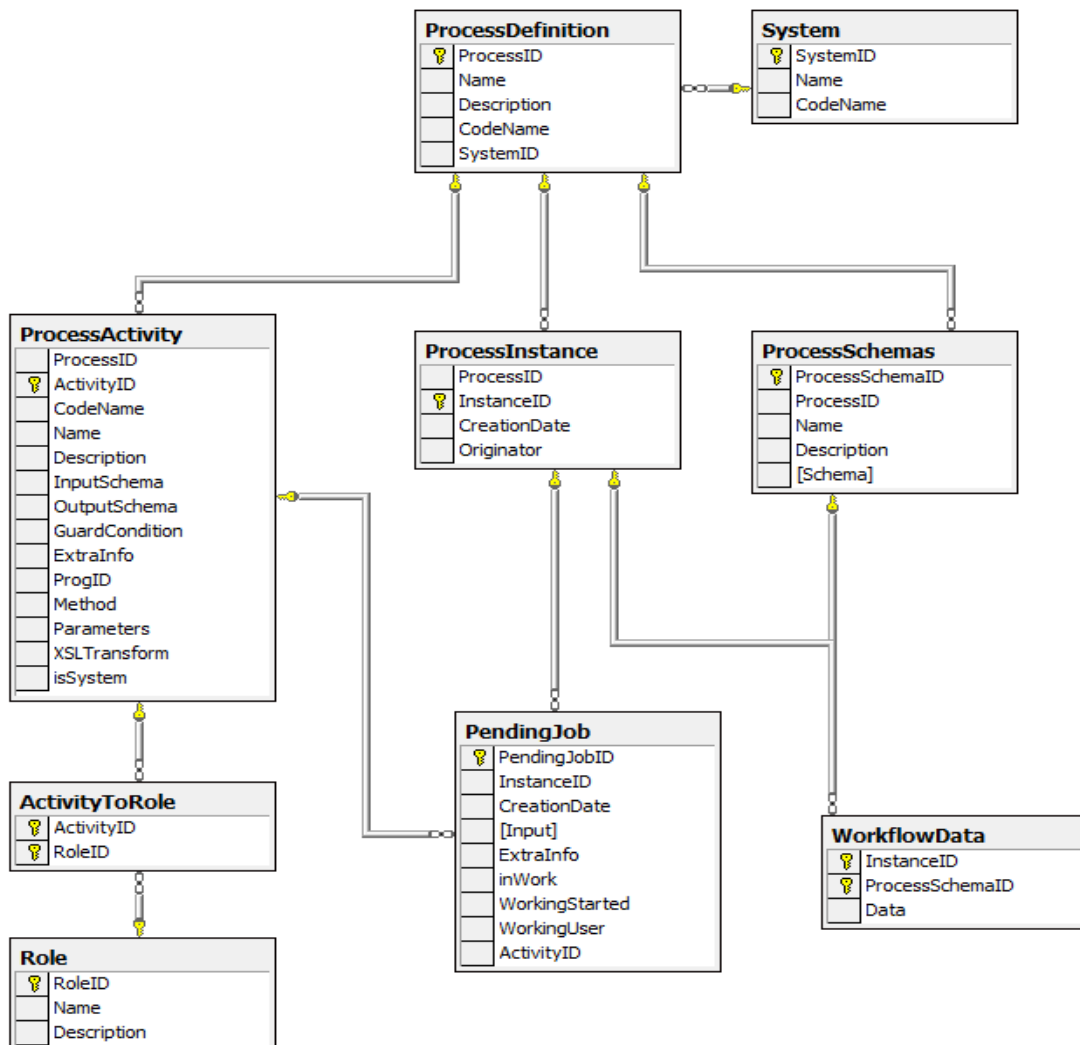


Figure 4.24: Database diagram of the implemented workflow inference engine

Finally, the implemented workflow inference engine runs as Component Object Model (COM+) software component under the Microsoft Transaction Server (MTS), which provides automated transaction and instance management, in order to ensure data consistency in the database in case transactions are not completed successfully and have to be rolled back.

4.6 An Information System based on SYMEX

As we mentioned in Section 4.3, SYMEX has been applied for modelling the business processes of the Accounting Office of Athens University of Economics and Business [182]. The information system has been implemented in 2004 and since then, it has been working and supporting real business needs. Next Sections discuss the business processes that have been modelled, the figures from 5 years of use, and the overall conclusions from the development of the System.

4.6.1 Process models

In order to model all the aspects of business that takes place within the Accounting Office, the development team ended with a number of 31 distinct SYMEX models and a total number of 267 Activities inside these models. The following table includes the list of modelled processes for the implemented Information System.

#	Process Codename	Modelled Activities	#	Process Codename	Modelled Activities
1	ACCOUNT_MG	2	17	INTERNALORD_MG	8
2	ASSET_MG	11	18	INTINVENTRAN_MG	8
3	AUDITTRAIL_MG	2	19	INVENTORY_MG	7
4	BNKACCOUNT_MG	11	20	OPENBAL_MG	7
5	BUDGET_MG	13	21	PAYORD_MG	21
6	BUDGETREVISION_MG	12	22	PAYREC_MG	13
7	BUDGETSCENARIO_MG	7	23	PREPAYORD_MG	11
8	CASHBALANCE_MG	7	24	PRQ_MG	9
9	CASHIER_MG	8	25	PSAACCOUNT_MG	7
10	CHECK_MG	11	26	PSAPARAMS_MG	5
11	CONSOLREL_MG	5	27	REPORT_MG	10
12	EGLSACCOUNT_MG	8	28	SPLR_MG	6
13	EGLSPARAMS_MG	12	29	SUPPLSRCDOC_MG	18
14	EGLSSTATISTIC_MG	2	30	UNITMEASURES_MG	6
15	EMPLOYEE_MG	7	31	WAREHOUSE_MG	7
16	EXTRAIT_MG	6		Total Activities	267

Table 4.2: List of modelled Processes

4.6.2 Figures

Table 4.3 has a number of interesting figures that we have extracted from the log files of the Information System.

Working Period	5 years and 10 months (January 2004 – now)
Number of users	17
Number of SYMEX models	31
Number of distinct Activities modelled	267
Number of distinct Activities executed	212 – 79,40% of total Activities modelled
Number of executions	478615
Number of errors in total executions	6266 – 1,3% of total executions

Table 4.3: Figures from the Information System based on SYMEX

As shown in table above, there are modelled Activities that have not yet executed within the Information System (22,60% - 55 out of 267). This is due to the modelling of Activities of the Payroll System of the University that the corresponding sub-department has not yet used. As far as the errors are concerned, from the 6266 recorded, 5423 of them were due to wrong user input. This led the development team to refine data validation rules in end user forms of the System, resulting to a number of only 37 error executions the last year. Moreover, the rest 843 errors were due to timeouts because of connection issues with the database.

4.6.3 Discussion

The modelling of Activities of the Accounting Office was done by two designers in cooperation with two members of the Office. The duration of process modelling kept about 3 months. During this phase, SYMEX helped designers and members of the Office to communicate their knowledge and proposals as the models proved easy to understand from people with no IT background and this has accelerated the modelling process. After then there was a testing period of 1 month where five members of the Accounting Office used the System simultaneously with the old one the Office had.

The Information System started working officially in January of 2004. The first 2 months, there was a need to adapt about 15 Activities based on the feedback we had from end users. Then in 2007, there was a need for a major redesign due to change of Greek legislation about purchases from Universities. The corresponding SYMEX model was adapted in two weeks time and successfully applied to the System without any side effects. Since then, the Information System works efficiently supporting all aspects of processes involved in the Accounting Office of the University.

Overall, the development team as well as the University's authorities are very satisfied from the result combined with the time and the cost needed for the project. We consider this as one of the toughest tests for the viability of SYMEX and we are glad that the outcome has mainly positive messages. Finally, the general conclusions about the advantages of SYMEX based on the experience from this implementation are the following:

1. Quick modelling of business processes
2. Easy communication of models with people having no IT background
3. Quick adaptation of modelled processes
4. Efficient execution

4.7 Conclusions

In this Chapter, we proposed an XML-based implementation approach in order to design and develop a working prototype of SYMEX. We encoded the concepts, properties, and relationships of SYMEX modelling in a top-down approach defining a Schema Definition (XSD). We chose XML because it can be easily processed by computers, and it is an extensible and open cross-platform W3C standard, endorsed by software industry market leaders [178-181]. Moreover, this XML-based approach could be easily supported by a graphical tool that would export the corresponding XML process descriptions of SYMEX models. On the other hand, XML may be considered as a strict and rudimentary language. It could turn as a disadvantage in case we would want to enrich our process descriptions with more semantics (such as OWL-s), in order to support automatic discovery and execution.

Apart from the implementation approach, we applied SYMEX modelling on a business process derived from the accounting domain. The example was used to demonstrate the basic methodology and principles of SYMEX modelling. Specifically, we showed that higher-levels of decomposition always depict high-level descriptions of business processes, while final levels of decomposition always encode the execution semantics of processes. One of the advantages of SYMEX modelling is that the visual models are straightforward and we argue that are easy to learn. On the other hand, there is no tool available yet for modelling business processes using SYMEX notation and exporting the corresponding XML process description. At the end, however easy and straightforward SYMEX modelling may be, it is a fact that most designers and developers use and trust well-established modelling approaches such as UML 2 Activity Diagrams and BPMN and are reluctant to try something new and not extensively tested.

This Chapter also presented a comparative analysis of change management between SYMEX models and UML 2 Activity Diagrams. The analysis confirmed that constraint-based modelling of SYMEX is more efficient in terms of adaptation in comparison with modelling with the construction of linear sequences of activities. Linear approach of modelling results in process designers needing to resolve all the sequence of activities and the dependencies between them in order to adapt processes to new business needs. This becomes a challenge when managing large and complex business processes, creating high maintenance costs. On the other hand, constraint-based reasoning of SYMEX models proves to be easier and quicker to adapt. In particular, SYMEX models do not provide any control flow constructs and they encode all business rules as constraints. This approach reduces the dependencies between Activities enabling much easier adaptation of the process. Process designers working with SYMEX models need to focus only on the change that needs to be done and not on the whole of the process.

Then, we presented the implementation approach for our workflow inference engine that executes SYMEX models. The algorithm of the engine is based on constraint-based theory and the execution of SYMEX models is a solution of a constraint-satisfaction problem; each Activity has a number of constraints that need to be satisfied in order to execute. Moreover, the implemented algorithm is not complex and can be easily implemented with any programming language. The first version of the engine was implemented using MS Visual Basic 6, but the code could be easily immigrated to Java [186] or .Net platform. Nevertheless, current implementation only supports the execution of components that are installed in one computer machine. In the next versions we plan to support web-service execution, as one of the main objectives of workflow technology is the distributed execution of business processes across organizational and platform boundaries using web-services. Finally, we discussed the implementation of the Information System in Athens University of Economics and Business, where SYMEX was applied as a modelling technique in order to capture all aspects of the Accounting Office's business processes.

5 Evaluation of the SYMEX Framework

5.1 Introduction

In Chapter 3, we introduced SYMEX and in Chapter 4, we assessed the frameworks applicability. In this Chapter, we evaluate our approach with respect to the complexity of SYMEX models, addressing research objective RO3. As we argued in Chapter 2, to the best of our knowledge, there is no integrated framework for modelling high-level process descriptions and low-level workflow execution semantics. However, three proposed mappings for modelling and execution exist, namely UML to BPEL4WS, BPMN to BPEL4WS and FBPML to OWL-s. Out of those three mappings, we consider the BPMN to BPEL4WS mapping to be the most significant and widely used, and therefore we are going to use it as the basis for comparison with our approach.

In order to prove that SYMEX is more efficient in producing easier to understand and maintain process models than the BPMN to BPEL4WS mapping, we have reviewed a number of complexity metrics for business processes. By applying such metrics, we can quantify the properties of SYMEX framework that we have informally described in previous Chapters, namely reduced complexity of the produced business process models that results in easier adaptation and maintainability. The complexity metric we eventually employ for the evaluation is the Cross-Connectivity metric, which can be applied to both BPMN and SYMEX models.

The structure of this Chapter is as follows. Section 5.2 discusses related research on metrics for business process models. Section 5.3 presents the Cross-Connectivity metric that we have selected for the evaluation. Section 5.4 describes the calculation steps of the metric and Section 5.5 applies the metric in a number of complexity cases in order to evaluate BPMN and SYMEX process models in terms of their complexity. Finally, Section 5.6 discusses the findings and summarises the work reported in this Chapter.

5.2 Measuring complexity of process models

Our research primarily deals with business process modelling. One of the main roles of business process models is to serve as a basis for communication between business and IT stakeholders. Therefore, process models should be easy to understand and maintain [187]. Complexity of business processes as defined by Cardoso [188] is: *"the degree to which a process is difficult to analyze, understand or explain. It may be characterized by the number and intricacy of activity interfaces, transitions, conditional and parallel branches, the existence of loops, data-flow, control-flow, roles, activity categories, the types of data structures, and other process characteristics."*

High complexity has undesirable effects on, among others, the validation, maintainability, and understandability of business process models [189]. Moreover, high complexity can lead to more errors and defects in development and testing and prohibit flexibility [190]. Therefore, the need for measures that can quantify the complexity of business process models is evident. Although hundreds of software complexity measures, named 'metrics', have been proposed in the last few decades, measuring the complexity of business process models is a rather new research with only a small number of contributions [187].

The early research on process model metrics was greatly inspired by software quality metrics [191], as business process and software program designs have many similarities [189]. Reijers and Vanderfeesten inspired from software metrics define cohesion and coupling metrics [192], for the design of activities in a workflow design. In [189] metrics similar to the Line-of-Code, McCabe's Cyclomatic Complexity, Halstead Complexity Metric and Information Flow Complexity are proposed to measure process complexity. Cardoso in [193] suggests a complexity metric, which is a generalization of McCabe's cyclomatic number, called the Control-Flow Complexity (CFC) metric. CFC is based on the analysis of XOR-splits, OR-splits and, AND-splits control statements in a process model. The main idea behind the CFC metric is the number of mental states that a designer has to consider when developing a process. Cardoso in [194] proposes, also another CFC metric to be used during the design of BPEL4WS processes, by calculating the complexity of the various types of workflow patterns encountered in the process. Finally, Lassen and Aalst in [195] extend the CFC metric defined by Cardoso and extend the Cyclomatic metric of McCabe. Table 5.1 shows a number of complexity metrics for software programs and their adaptation for business process models [196].

Software complexity metric	Corresponding metric for business process model	Usage, significance
Lines of Code	Number of activities (and connectors)	Very simple, does not take into account the control-flow
Cyclomatic number	CFC as defined by Cardoso	Measures the number of possible control flow decisions, well-suited for measuring the number of test cases needed to test the model, does not take into account other structure-related information
Nesting depth	Nesting depth	Provides information about structure
Knot-count	Counting the "reasons for unstructuredness"	Measure of "well-structuredness" (for example jumps out of or into control-flow structures) Is always 0 for well-structured models
Cognitive weight	Cognitive weight	Measures the cognitive effort to understand a model, can indicate that a model should be re-designed
(Anti)Patterns	(Anti)Patterns for business process models	Experience with the patterns needed. Counting the usage of anti-patterns in a model can help to detect poor modelling
Fan-in/Fan-out	Fan-in/Fan-out	Can indicate poor modularization

Table 5.1: Complexity metrics for software and business process models [196]

Apart from the domain of software, metrics for process modelling have been inspired from other domains too. In [189] graph theoretical measures are proposed as potential complexity metrics for business process models. [197] proposes a density metric inspired by social network analysis in order to quantify the complexity of EPC models. [198] presents a metric calculating the Log-Based Complexity (LBC) of a workflow process, by devising complexity metrics for each workflow pattern identified in log traces generated from the execution of the workflow. Finally, [191] introduces the cross-connectivity metric that builds on insights from cognitive research. The next Section describes in detail the cross-connectivity metric, as this has been selected as the basis for evaluating the SYMEX framework.

5.3 Cross-connectivity metric (CC)

5.3.1 Overview

As argued in the previous Section, a business process model should be capable to serve as a basis for communication between the people involved in a software development process. This is similar to an architect using a model to ascertain the views of users, to communicate new ideas and develop a shared understanding among participants [191]. Therefore, it is crucial to have process models that are easily understood by people. Although there is some research that identifies the cognitive motivation as a potential inspiration for metrics [189], up to now there has been little research on measurements to quantify the cognitive effort of a model reader to understand a process model. One of the few examples is the research on the Control Flow Complexity (CFC) metric by Cardoso and the Cross-Connectivity (CC) metric [191].

Cross-Connectivity (CC) metric was recently proposed by Vanderfeesten et al. [191] and is based on cognitive research in programming languages. The authors argue that the study of a process model requires some hard mental operations and involves behavioural relationships between model elements being constructed in the mind of the user. Understanding of issues such as the dependencies between the elements of a process and whether pairs of activities with lots of parallelism and choices are exclusive or not, are very cognitively demanding tasks even for experts. The CC metric aims to quantify the ease of understanding in terms of the interplay of any pair of model elements. Vanderfeesten et al. evaluate and validate the metric using a thorough empirical evaluation, in order to establish its suitability for assessing the ease of understanding of process models.

The reasons for selecting the CC metric, out of all the metrics found in the literature, are the following:

1. CC metric does not take into account only control-flow elements (splits) such as other metrics, e.g. CFC.
2. CC metric is based on cognitive research, quantifying the ease of understanding a process model from the user's perspective. It adds a new cognitive perspective on process model quality [191].
3. CC metric takes into account all characteristics of a process, namely tasks, joins, arcs, paths, number of tasks.
4. CC metric has a solid evaluation and validation [191].
5. CC metric is independent of syntactic features and can be applied to standard modelling languages such as EPCs, UML Activity Diagrams, Petri nets, BPMN, YAWL etc. [191]. Correspondingly, it can also be applied to SYMEX process models.

6. CC metric provides a common base for comparing two or more modelling approaches. Higher scores from application of the metric to a process model are associated with an easier understanding of the model [191].

5.3.2 Formulas

There are four assumptions used for the formalization of the metric [191]:

1. The CC metric expresses the sum of the paths between all pairs of nodes in a process model, relative to the theoretical maximum number of paths between all nodes (see Definition 5).
2. The path with the highest connectivity between two nodes determines the strength of the overall connectivity between those nodes (see Definition 4).
3. The tightness of a path (i.e., degree of connectivity) is determined by the product of the valuations of the links connecting the nodes on the path (see Definition 3).
4. Differences in the types of nodes that a path consists of determine the tightness of the arcs connecting nodes (see Definitions 1 and 2).

Five definitions as given in [191] are as follows:

Definition 1 (Weight of a Node).

Let a process model be given as a graph consisting of a set of nodes ($n_1, n_2, \dots \in N$) and a set of directed arcs ($a_1, a_2, \dots \in A$). A node can be of one of two types: (i) task, e.g. $t_1, t_2 \in T$, and (ii) connector, e.g. $c_1, c_2 \in C$. Thus, $N = T \cup C$.

The weight of a node n , $w(n)$, is defined as follows:

$$w(n) = \begin{cases} 1 & , \text{ if } n \in C \wedge n \text{ is of type AND} \\ \frac{1}{d} & , \text{ if } n \in C \wedge n \text{ is of type XOR} \\ \frac{1}{2^d - 1} + \frac{2^d - 2}{2^d - 1} \cdot \frac{1}{d} & , \text{ if } n \in C \wedge n \text{ is of type OR} \\ 1 & , \text{ if } n \in T \end{cases} ,$$

with d the degree of the node (i.e. the total number of ingoing and outgoing arcs of the node). Definition 1 assumes that the process model consists of tasks and connectors, where tasks have at most one input and one output arc and connectors can have multiple input and output arcs.

Definition 2 (Weight of an Arc).

Let a process model be given by a set of nodes (N) and a set of directed arcs (A). Each directed arc (a) has a source node (denoted by $src(a)$) and a destination node (denoted by $dest(a)$). The weight of arc a , $W(a)$, is defined as follows:

$$W(a) = w(src(a)) \cdot w(dest(a))$$

Definition 3 (Value of a Path).

Let a process model be given by a set of nodes (N) and a set of directed arcs (A). A path p from node n_1 to node n_2 is given by the sequence of directed arcs that should be followed from n_1 to n_2 : $p = \langle a_1, a_2, \dots, a_x \rangle$. The value for a path p , $v(p)$, is the product of the weights of all arcs in the path:

$$v(p) = W(a_1) \cdot W(a_2) \cdot \dots \cdot W(a_x)$$

Definition 4 (Value of a Connection).

Let a process model be given by a set of nodes (N) and a set of directed arcs (A) and let P_{n_1, n_2} be the set of paths from node n_1 to n_2 . The value of the connection from n_1 to n_2 , $V(n_1, n_2)$, is the maximum value of all paths connecting n_1 and n_2 :

$$V(n_1, n_2) = \max_{p \in P_{n_1, n_2}} v(p)$$

If no path exists between node n_1 and n_2 , then $V(n_1, n_2) = 0$. Also loops in a path should not be considered more than once, since the value of the connection will not be higher if the loop is followed more than once in the particular path.

Definition 5 (Cross-Connectivity (CC)).

Let a process model be given by a set of nodes (N) and a set of directed arcs (A). The Cross-Connectivity metric is then defined as follows:

$$CC = \frac{\sum_{n1, n2 \in N} V(n1, n2)}{|N| \cdot (|N| - 1)}$$

As for the interpretation of the value, a higher value of the metric is associated with an easier understanding of the model, which implies consequently a lower error probability.

5.3.3 Adapting CC metric to SYMEX process models

As we explained in Chapter 3, SYMEX process models have the following key constructs:

1. Activities, which are equivalent to nodes of type 'task'.
2. Inputs and Outputs, which can be considered as the directed arcs of the graph of the process model.
3. Controls, which are integrated part of Activities and can be considered as equivalent to logical connectors (AND, OR, XOR).
4. Execution Mechanisms, which are part of Activities.

Therefore, the CC metric can be applied to SYMEX process models under the following considerations:

1. There are no nodes in SYMEX models acting as connectors.
2. SYMEX process models have Activities with more than one Input, while Definition 1 of CC metric notes that each task must have at most one input.
3. SYMEX process models support hierarchy through the decomposition of composite Activities and CC metric does not take into account decomposition of process models.
4. CC metric does not have a concept equivalent to the Execution Mechanisms of a SYMEX process model.

The first three points above are crucial in order to evaluate the complexity of a SYMEX process model. Consequently, these three points will be used for adapting the formulas of the CC metric. Regarding the fourth point above, Execution Mechanisms encode the means by which Activities are executed and thus are effectively software components, web-services or business roles responsible for the realization of Activities. For that reason, we consider them to have minor influence in the understanding of a model by a user and therefore they will not be included in the adapted CC metric.

The original CC metric is adapted as follows. Original definitions 2, 3 and 4 remain the same, while definitions 1 and 5 have the following changes:

Adapted Definition 1 (Weight of a Node).

Let a process model be given as a graph consisting of a set of nodes ($n_1, n_2, \dots \in N$) and a set of directed arcs ($a_1, a_2, \dots \in A$). A node can be of one of two types: (i) atomic Activity, e.g. $AtAct_1, \dots, AtAct_2 \in AtAct$, and (ii) composite Activity, e.g. $CAct_1, \dots, CAct_2 \in CAct$, with a composite Activity $CAct_i$ having: (a) a set of nodes ($n_1, n_2, \dots \in N(CAct_i)$) at its decomposition model and (b) a CC metric value of

$CC_metric(CAct_i)$. Thus, $N = AtAct \cup CAct$ and the weight of a node n , $w(n)$, is defined as follows:

$$w(n) = \begin{cases} 1 & , \text{ if } n \in AtAct \wedge d=2 \\ \frac{1}{2^d-1} + \frac{2^d-2}{2^d-1} \cdot \frac{1}{d} & , \text{ if } n \in AtAct \wedge d>2 \\ CC_metric(n) & , \text{ if } n \in CAct \end{cases} ,$$

with d being the degree of the node (i.e. the total number of ingoing and outgoing arcs of the node). Based on the formalization of SYMEX in Chapter 3, an atomic Activity in a SYMEX process model has always exactly one Output. Based on the above definition, we have two cases of atomic Activities:

- If $d=2$, the Activity has exactly one Input and one Output. In this case, Activity is equivalent to the concept of a task which, based on the original Definition 1, has at most one input and one output arc and its weight is 1. So, $w(n)=1$.
- If $d>2$, the Activity has more than one Inputs. In this case we don't know upfront how many of the Inputs will be followed. It could be that they are all followed (cf. AND situation), that only one Input is followed (cf. XOR situation), but it could also well be that several Inputs are followed. So the weight of the Activity in this case is equal to the weight of an OR connector [191].

Definition 5 (Cross-Connectivity (CC)).

The Cross-Connectivity metric is defined as:

$$CC = \frac{\sum_{n1, n2 \in N} V(n1, n2)}{|N| \cdot (|N| - 1)} ,$$

where $|N|$ is the number of nodes of the model. In SYMEX models, we have two kinds of nodes, namely atomic Activities and composite Activities. As we defined in previous definition, a composite Activity $CAct_i$ has a set of nodes ($n1, n2, \dots \in N(CAct_i)$) at its decomposition model. Therefore, the number of nodes at a level of a model is defined as:

$$NodesNumber = |AtAct| + \sum_{\forall CAct_i \in CAct, n_i \in N(CAct_i)} \frac{1}{|n_i|}$$

The final formula for the adapted CC metric is:

$$CC = \frac{\sum_{n1, n2 \in N} V(n1, n2)}{NodesNumber \cdot (NodesNumber - 1)}$$

In hierarchical models, users typically study the model in a top-down fashion; users first study a high-level model and then follow composite Activities that lead to decomposed models of more detail. However, in order to calculate the weight of SYMEX hierarchical models we follow a bottom-up approach. First, we evaluate the CC metric for the lowest levels of decomposition. Then we assign the value of the CC metric to the composite Activity that decomposes to that model and we continue till we reach the highest level of the model. Figure 5.1 shows the top-down approach a user follows to read a hierarchical SYMEX model in comparison with the bottom-up approach we follow to calculate the CC metric for this model.

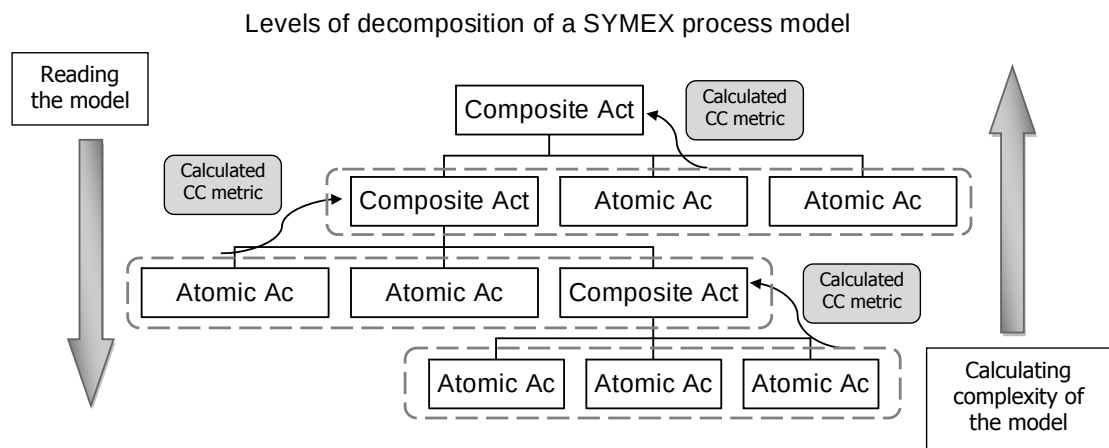


Figure 5.1: Reading a SYMEX model and calculating the CC metric

5.4 Calculation of metric

In this Section we apply the CC metric to a process model in BPMN and then to a 'flat' and a hierarchical SYMEX model. The purpose is to evaluate and compare the complexity of these models.

5.4.1 BPMN process model

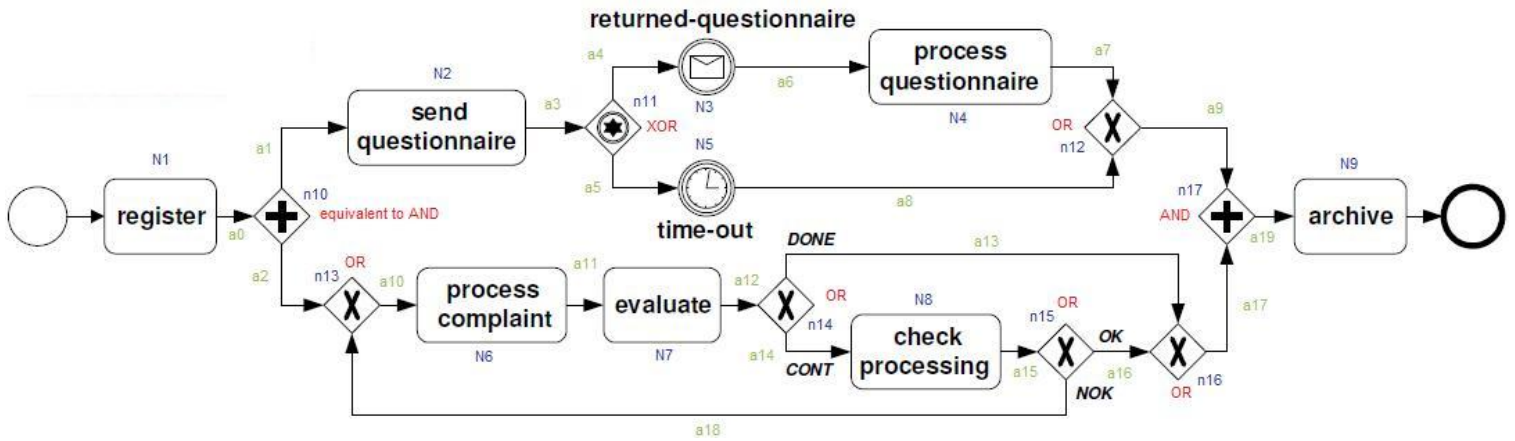


Figure 5.2: BPMN process model [49]

A business process model in BPMN is shown in Figure 5.2 and is described as follows [49]. When a complaint is registered, a questionnaire is sent to the complainant and, in parallel, the complaint is processed. If the complainant returns the questionnaire within two weeks, the task 'process questionnaire' is performed, otherwise the result of the questionnaire is discarded. In parallel, the complaint is evaluated. Based on the result, the evaluation either is completed or continues with the task 'check processing'. If the check processing result is not ok, the complaint requires re-processing. Finally, the complaint is archived.

For the calculation of the CC metrics, events of the BPMN process model are considered of equivalent weight to nodes of type task. Next, we calculate the weight of nodes, arcs, connections and in the end the CC metric, based on the formulas of Section 5.3.2.

5.4.1.1 Weight of nodes

Node (n)	Degree (m) (total number of ingoing and outgoing arcs of the node)	Weight W(n)
N1	1	1
N2	2	1
N3	2	1
N4	2	1
N5	2	1
N6	2	1
N7	2	1
N8	2	1
N9	1	1
n10 (AND)	3	1
n11 (XOR)	3	$\frac{1}{3}$
n12 (OR)	3	$\frac{1}{2^3-1} + \frac{2^3-2}{2^3-1} \cdot \frac{1}{3} = \frac{3}{7}$
n13 (OR)	3	$\frac{3}{7}$
n14 (OR)	3	$\frac{3}{7}$
n15 (OR)	3	$\frac{3}{7}$
n16 (OR)	3	$\frac{3}{7}$
n17 (AND)	3	$\frac{3}{7}$

Table 5.2: Calculation of weight of nodes for BPMN process model

5.4.1.2 Weight of arcs

Arc (a)	Source node src(a)	Destination node dest(a)	Weight $W(a)=W(\text{src}(a)) * W(\text{dest}(a))$
a0	N1	n10	$1 \cdot 1 = 1$
a1	n10	N2	$1 \cdot 1 = 1$
a2	n10	n13	$1 \cdot \frac{3}{7} = \frac{3}{7}$
a3	N2	n11	$1 \cdot \frac{1}{3} = \frac{1}{3}$
a4	n11	N3	$\frac{1}{3} \cdot 1 = \frac{1}{3}$
a5	n11	N5	$\frac{1}{3} \cdot 1 = \frac{1}{3}$
a6	N3	N4	$1 \cdot 1 = 1$
a7	N4	n12	$1 \cdot \frac{3}{7} = \frac{3}{7}$
a8	N5	n12	$1 \cdot \frac{3}{7} = \frac{3}{7}$
a9	n12	n17	$\frac{3}{7} \cdot 1 = \frac{3}{7}$
a10	n13	N6	$\frac{3}{7} \cdot 1 = \frac{3}{7}$
a11	N6	N7	$1 \cdot 1 = 1$
a12	N7	n14	$1 \cdot \frac{3}{7} = \frac{3}{7}$
a13	n14	n16	$\frac{3}{7} \cdot \frac{3}{7} = \frac{9}{49}$
a14	n14	N8	$\frac{3}{7} \cdot 1 = \frac{3}{7}$
a15	N8	n15	$1 \cdot \frac{3}{7} = \frac{3}{7}$
a16	n15	n16	$\frac{3}{7} \cdot \frac{3}{7} = \frac{9}{49}$
a17	n16	n17	$\frac{3}{7} \cdot 1 = \frac{3}{7}$
a18	n15	n13	$\frac{3}{7} \cdot \frac{3}{7} = \frac{9}{49}$
a19	n17	N9	$1 \cdot 1 = 1$

Table 5.3: Calculation of weight of arcs for BPMN process model

5.4.1.3 Weight of connections

	N1	N2	N3	N4	N5	N6	N7	N8	N9	n10	n11	n12	n13	n14	n15	n16	n17	Total
N1	0	1	1/9	1/9	1/9	9/49	9/49	14/415	1/49	1	1/3	1/21	3/7	27/343	6/415	6/415	1/49	3 646/933
N2	0	0	1/9	1/9	1/9	0	0	0	1/49	0	1/3	1/21	0	0	0	0	1/49	37/49
N3	0	0	0	1	0	0	0	0	9/49	0	0	3/7	0	0	0	0	9/49	1 39/49
N4	0	0	0	0	0	0	0	0	9/49	0	0	3/7	0	0	0	0	9/49	39/49
N5	0	0	0	0	0	0	0	0	9/49	0	0	3/7	0	0	0	0	9/49	39/49
N6	0	0	0	0	0	0	1	9/49	14/415	0	0	0	6/415	3/7	27/343	27/343	14/415	1 132/155
N7	0	0	0	0	0	5/807	0	9/49	14/415	0	0	0	6/415	3/7	27/343	27/343	14/415	555/647
N8	0	0	0	0	0	14/415	14/415	0	14/415	0	0	0	27/343	6/415	3/7	27/343	14/415	189/257
N9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n10	0	1	1/9	1/9	1/9	9/49	9/49	14/415	1/49	0	1/3	1/21	3/7	27/343	6/415	6/415	1/49	2 646/933
n11	0	0	1/3	1/3	1/3	0	0	0	3/49	0	0	1/7	0	0	0	0	3/49	1 13/49
n12	0	0	0	0	0	0	0	0	3/7	0	0	0	0	0	0	0	3/7	6/7
n13	0	0	0	0	0	3/7	3/7	27/343	6/415	0	0	0	0	9/49	14/415	14/415	6/415	1 198/917
n14	0	0	0	0	0	6/415	6/415	3/7	27/343	0	0	0	14/415	0	9/49	9/49	27/343	1 2/125
n15	0	0	0	0	0	27/343	27/343	6/415	27/343	0	0	0	9/49	14/415	0	9/49	27/343	233/319
n16	0	0	0	0	0	0	0	0	3/7	0	0	0	0	0	0	0	3/7	6/7
n17	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1

Table 5.4: Calculation of weight of connections for BPMN process model

5.4.1.4 CC metric

CC $\approx$ 0,0769

5.4.2 SYMEX process model without decomposition

In this Section, we model the same process in only one level without any decomposition ('flat' model). In this case, the model consists only of atomic Activities. Figure 5.3 shows the model.

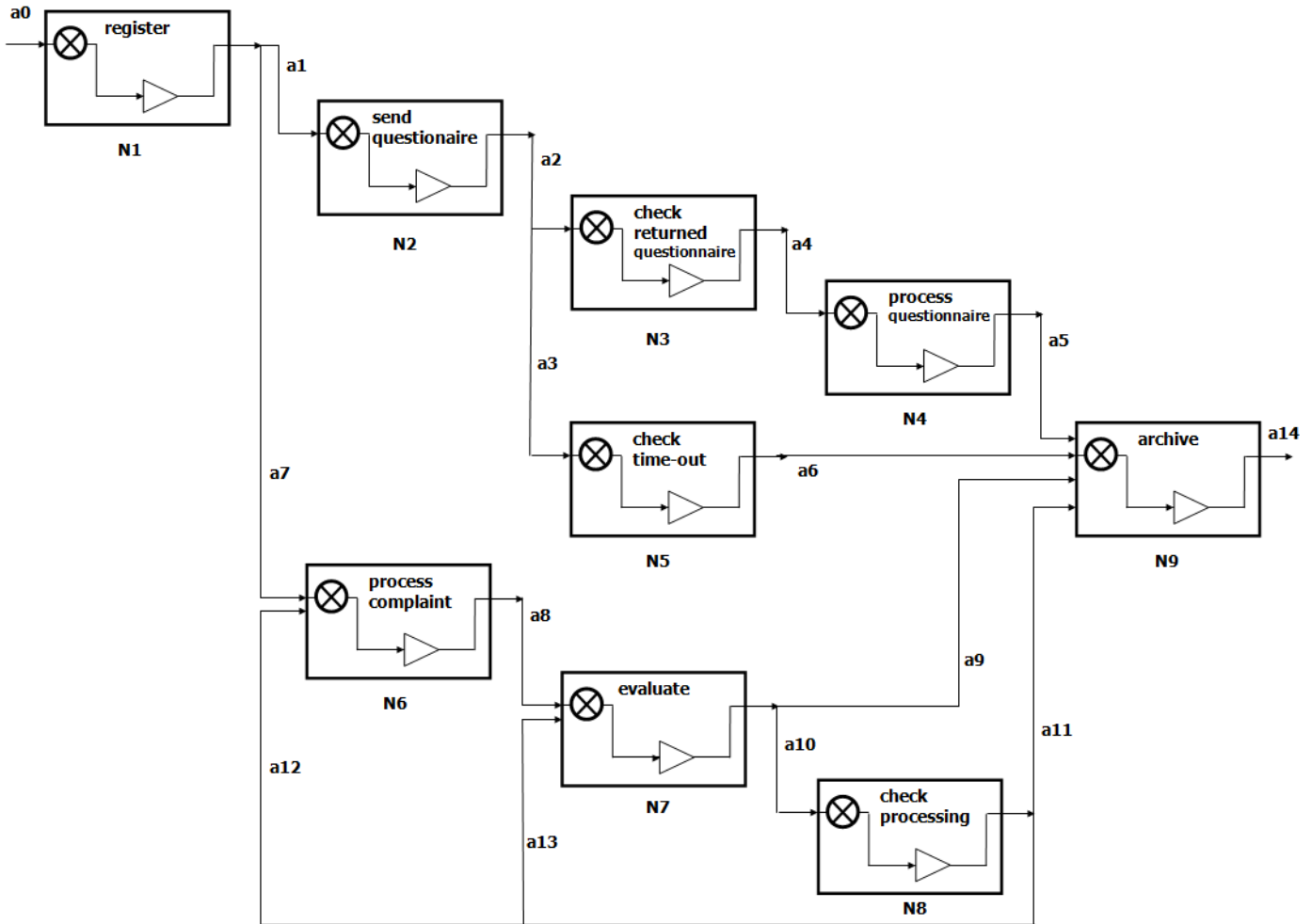


Figure 5.3: SYMEX model without decomposition

The calculations of the weight of nodes, arcs, connections and of the CC metric that follow are based on the adapted formulas of Section 5.3.3.

5.4.2.1 Weight of nodes

Node (n)	Degree (m) (total number of ingoing and outgoing arcs of the node)	Weight W(n)
N1	2	1
N2	2	1
N3	2	1
N4	2	1
N5	2	1
N6	3	$\frac{1}{2^3-1} + \frac{2^3-2}{2^3-1} \cdot \frac{1}{3} = \frac{3}{7}$
N7	3	$\frac{3}{7}$
N8	2	1
N9	5	$\frac{1}{2^5-1} + \frac{2^5-2}{2^5-1} \cdot \frac{1}{5} = \frac{7}{31}$

Table 5.5: Calculation of weight of nodes for SYMEX process model

5.4.2.2 Weight of arcs

Arc (a)	Source node src(a)	Destination node dest(a)	Weight W(a)=W(src(a)) * W(dest(a))
a1	N1	N2	$1 \cdot 1 = 1$
a2	N2	N3	$1 \cdot 1 = 1$
a3	N2	N5	$1 \cdot 1 = 1$
a4	N3	N4	$1 \cdot 1 = 1$
a5	N4	N9	$1 \cdot \frac{7}{31} = \frac{7}{31}$
a6	N5	N9	$1 \cdot \frac{7}{31} = \frac{7}{31}$
a7	N1	N6	$1 \cdot \frac{3}{7} = \frac{3}{7}$
a8	N6	N7	$\frac{3}{7} \cdot \frac{3}{7} = \frac{9}{49}$
a9	N7	N9	$\frac{3}{7} \cdot \frac{7}{31} = \frac{3}{31}$
a10	N7	N8	$\frac{3}{7} \cdot 1 = \frac{3}{7}$
a11	N8	N9	$1 \cdot \frac{7}{31} = \frac{7}{31}$
a12	N8	N6	$1 \cdot \frac{3}{7} = \frac{3}{7}$
a13	N8	N7	$1 \cdot \frac{3}{7} = \frac{3}{7}$

Table 5.6: Calculation of weight of arcs for SYMEX process model

5.4.2.3 Weight of connections

	N1	N2	N3	N4	N5	N6	N7	N8	N9	Total
N1	0	1	1	1	1	3/7	27/343	14/415	7/31	4 467/609
N2	0	0	1	1	1	0	0	0	7/31	3 7/31
N3	0	0	0	1	0	0	0	0	7/31	1 7/31
N4	0	0	0	0	0	0	0	0	7/31	7/31
N5	0	0	0	0	0	0	0	0	7/31	7/31
N6	0	0	0	0	0	0	9/49	27/343	4/225	202/721
N7	0	0	0	0	0	9/49	0	3/7	3/31	173/244
N8	0	0	0	0	0	3/7	27/343	0	7/31	206/281
N9	0	0	0	0	0	0	0	0	0	0

Table 5.7: Calculation of weight of connections for SYMEX process model

5.4.2.4 CC metric

The number of nodes is:

$$NodesNumber = |AtAct| + \sum_{\forall CAct_i \in CAct, n_i \in N(CAct_i)} \frac{1}{|n_i|} = |AtAct| = 9$$

Therefore, the value of the CC metric is:

$$CC = \frac{\sum_{n1, n2 \in N} V(n1, n2)}{NodesNumber \cdot (NodesNumber - 1)} = \frac{11 \frac{215}{548}}{9 \cdot 8} \approx 0,1581$$

The calculated value is greater than that of the BPMN model, meaning that the 'flat' SYMEX model is easier to understand than the equivalent BPMN model.

5.4.3 SYMEX process model with decomposition

In this Section, we model the same process in more levels of detail. Here the model consists of a first level model and two decompositions of the composite Activities. Figure 5.4 shows Level 1 of the model.

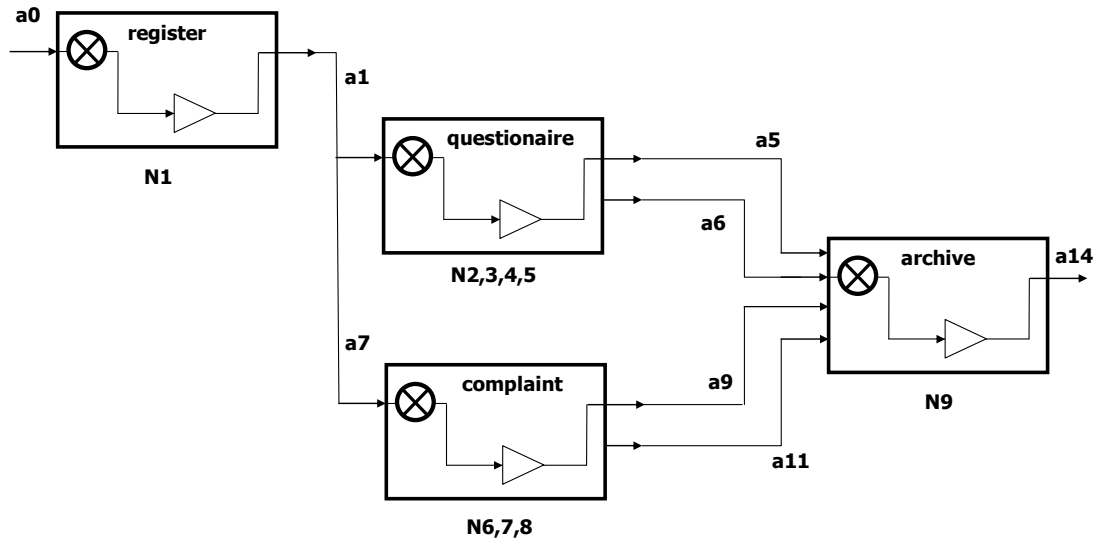


Figure 5.4: Level 1 of SYMEX model

Figure 5.5 shows the decomposition of composite Activity “questionnaire”.

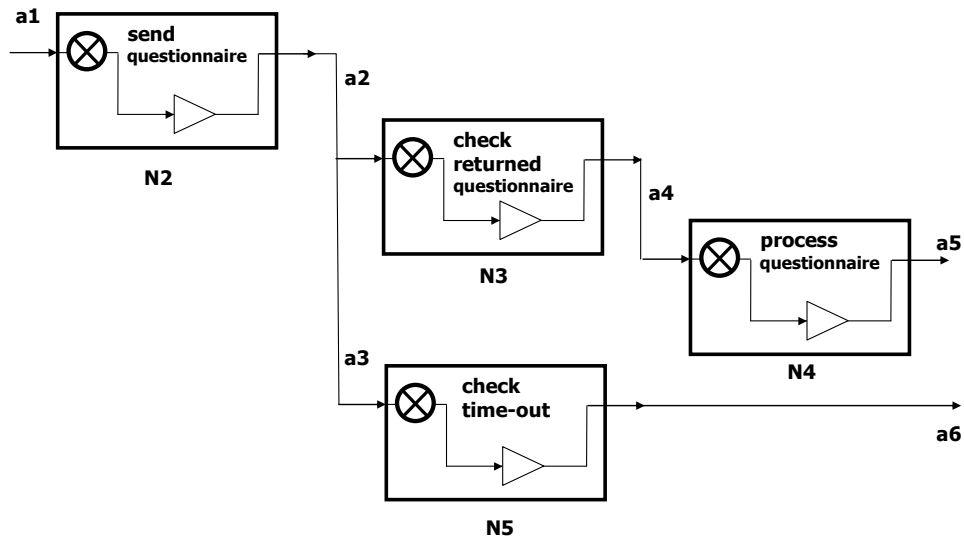


Figure 5.5: Level 2 of SYMEX model: decomposed Activity “questionnaire”

Figure 5.6 shows the decomposition of composite Activity “complaint”.

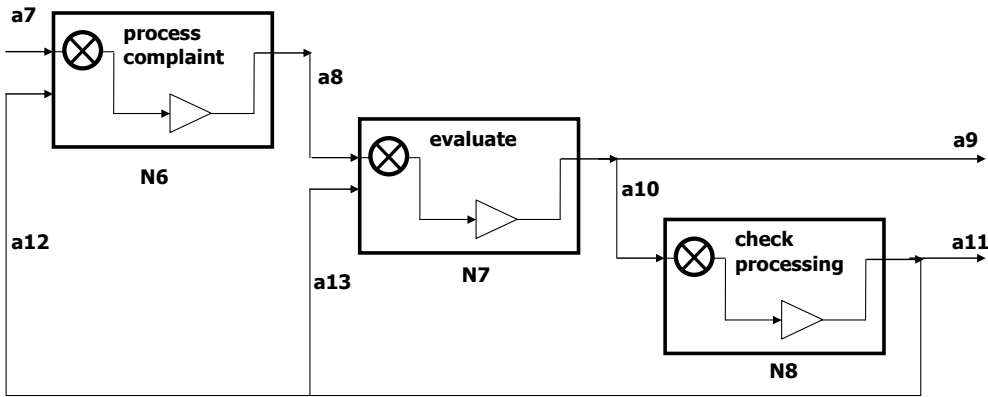


Figure 5.6: Level 2 of SYMEX model: decomposed Activity "complaint"

The order of carrying out the calculations is as follows. First, we calculate the CC metric for the decomposed models of Figure 5.5 and Figure 5.6 and then we calculate the CC metric of Level 1 of the SYMEX model of Figure 5.4. The calculations of the weights of nodes, arcs, connections and of the CC metric that follow are based on the adapted formulas of Section 5.3.3.

5.4.3.1 Decomposition of Activity N2,3,4,5

The calculation of the CC metric begins with the models of Figure 5.5 and Figure 5.6, as they are composed by atomic Activities only. The weight of nodes, arcs, and connections for the models, are shown in the tables below.

Weight of nodes

Node (n)	Degree (m)	Weight W(n)
N2	2	1
N3	2	1
N4	2	1
N5	2	1

Table 5.8: Calculation of weight of nodes for Activity N2,3,4,5 of SYMEX model

Weight of arcs

Arc (a)	Source node src(a)	Destination node dest(a)	Weight $W(a) = W(\text{src}(a)) * W(\text{dest}(a))$
a2	N2	N3	$1 \cdot 1 = 1$
a3	N2	N5	$1 \cdot 1 = 1$
a4	N3	N4	$1 \cdot 1 = 1$

Table 5.9: Calculation of weight of arcs for Activity N2,3,4,5 of SYMEX model

Weight of connections

	N2	N3	N4	N5	Total
N2	0	1	1	1	3
N3	0	0	1	0	1
N4	0	0	0	0	0
N5	0	0	0	0	0

Table 5.10: Calculation of weight of connections for Activity N2,3,4,5 of SYMEX model**CC metric**

$$CC = \frac{4}{4 \cdot 3} = \frac{1}{3}$$

5.4.3.2 Decomposition of Activity N6,7,8**Weight of nodes**

Node (n)	Degree (m)	Weight W(n)
N6	3	$\frac{1}{2^3-1} + \frac{2^3-2}{2^3-1} \cdot \frac{1}{3} = \frac{3}{7}$
N7	3	$\frac{3}{7}$
N8	2	1

Table 5.11: Calculation of weight of nodes for Activity N6,7,8 of SYMEX model**Weight of arcs**

Arc (a)	Source node src(a)	Destination node dest(a)	Weight $W(a)=W(\text{src}(a)) * W(\text{dest}(a))$
a8	N6	N7	$\frac{3}{7} \cdot \frac{3}{7} = \frac{9}{49}$
a10	N7	N8	$\frac{3}{7} \cdot 1 = \frac{3}{7}$
a12	N8	N6	$1 \cdot \frac{3}{7} = \frac{3}{7}$
a13	N8	N7	$1 \cdot \frac{3}{7} = \frac{3}{7}$

Table 5.12: Calculation of weight of arcs for Activity N6,7,8 of SYMEX model

Weight of connections

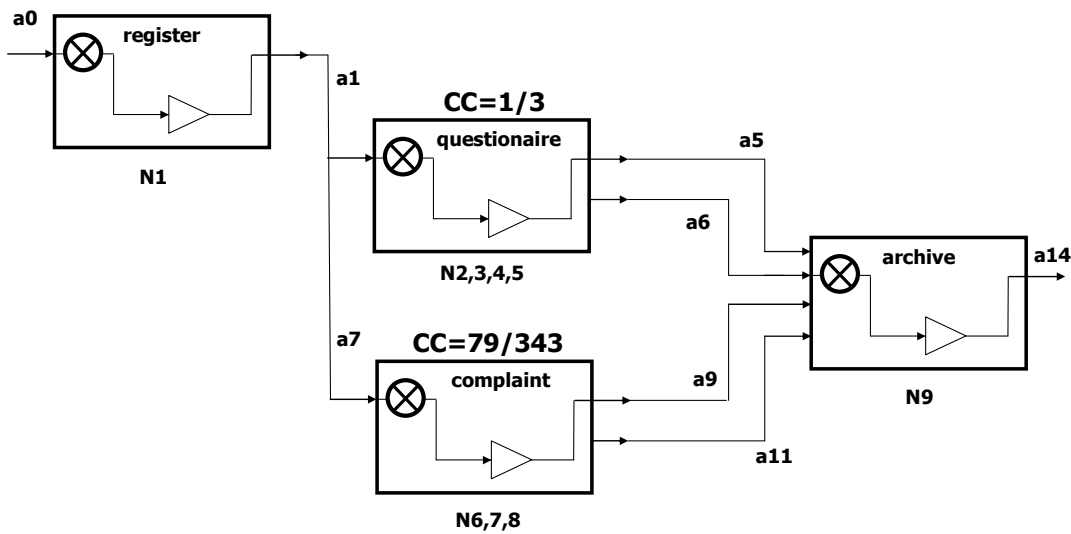
	N6	N7	N8	Total
N6	0	9/49	27/343	90/343
N7	9/49	0	3/7	30/49
N8	3/7	27/343	0	174/343

Table 5.13: Calculation of weight of connections for Activity N6,7,8 of SYMEX model**CC metric**

$$CC = \frac{\frac{474}{343}}{3 \cdot 2} = \frac{79}{343}$$

5.4.3.3 Level 1 of SYMEX model

Now that we have calculated the CC metric for the decomposed Activities, we calculate the CC metric for the first level model, as shown in Figure 5.7.

**Figure 5.7: Level 1 of SYMEX model with calculated CC metric for the composite Activities**

The weight of nodes, arcs, and connections for the model, are shown in the tables below.

Weight of nodes

Node (n)	Degree (m)	Weight W(n)
N1	2	1
N2,3,4,5	--- composite Activity ---	1/3
N6,7,8	--- composite Activity ---	79/343
N9	5	$\frac{1}{2^5-1} + \frac{2^5-2}{2^5-1} \cdot \frac{1}{5} = \frac{7}{31}$

Table 5.14: Calculation of weight of nodes for Level 1 of SYMEX model**Weight of arcs**

Arc (a)	Source node src(a)	Destination node dest(a)	Weight $W(a)=W(src(a)) * W(dest(a))$
a1	N1	N2,3,4,5	$1 \cdot \frac{1}{3} = \frac{1}{3}$
a5	N2,3,4,5	N9	$\frac{1}{3} \cdot \frac{7}{31} = \frac{7}{93}$
a6	N2,3,4,5	N9	$\frac{1}{3} \cdot \frac{7}{31} = \frac{7}{93}$
a7	N1	N6,7,8	$1 \cdot \frac{79}{343} = \frac{79}{343}$
a9	N6,7,8	N9	$\frac{79}{343} \cdot \frac{7}{31} = \frac{22}{423}$
a11	N6,7,8	N9	$\frac{79}{343} \cdot \frac{7}{31} = \frac{22}{423}$

Table 5.15: Calculation of weight of arcs for Level 1 of SYMEX model**Weight of connections**

	N1	N2,3,4,5	N6,7,8	N9	Total
N1	0	1/3	79/343	7/279	136/231
N2,3,4,5	0	0	0	7/93	7/93
N6,7,8	0	0	0	22/423	22/423
N9	0	0	0	0	0

Table 5.16: Calculation of weight of connections for Level 1 of SYMEX model

CC metric

The number of nodes is:

$$NodesNumber = |AtAct| + \sum_{\forall CAct_i \in CAct, n_i \in N(CAct_i)} \frac{1}{|n_i|} = 2 + \frac{1}{4} + \frac{1}{3} = 2\frac{7}{12}$$

Therefore, the value of the CC metric is:

$$CC = \frac{\sum_{n1, n2 \in N} V(n1, n2)}{NodesNumber \cdot (NodesNumber - 1)} = \frac{0,716}{2\frac{7}{12} (2\frac{7}{12} - 1)} \approx 0,175$$

The calculated value is greater than the value of the 'flat' SYMEX model and is greater than the value of the BPMN model. Thus, it appears that the hierarchical SYMEX model is easier to understand than both the 'flat' SYMEX model and the equivalent BPMN model.

5.5 Complexity Cases

This Section reports four process models where we have calculated the CC complexity metric. The focus is on the breadth of the cases that demonstrate complexity difference between BPMN and SYMEX models and not on the calculations of each case.

5.5.1 Trouble Ticket Process

5.5.1.1 Process Description

"This is a process to allow a software company to handle a customer support issue. This issue might be handled by the customer support team directly. If not, it is forwarded to the QA team for validation/verification and possibly handling the issue there. If it is a real problem in the product, it is referred to the development team for a fix. Before the process is complete, there are steps to assure that the customer who reported the problem gets the final resolution." [199]

5.5.1.2 BPMN Model

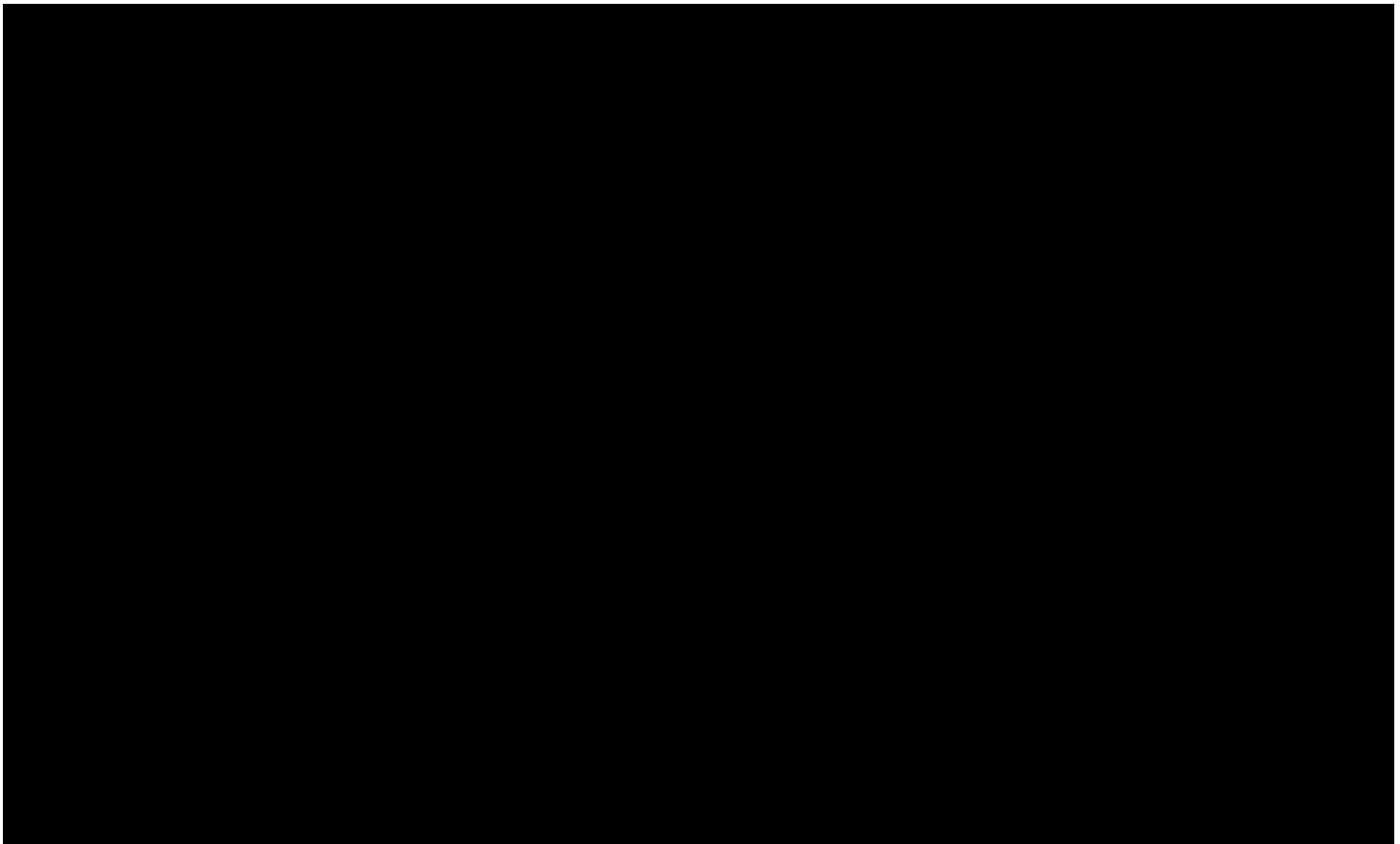


Figure 5.8: BPMN model: Trouble Ticket Process [199]

5.5.2 Supply Fulfilment Process

5.5.2.1 Process Description

This is a typical process of a company in order to get supplies for the warehouse. When there is an order request from a customer, there is a check whether the requested item is in or out of stock. In case it is in stock, the item is delivered to the customer and the process ends. On the other hand, when it is out of stock, the company sends request for offers to the suppliers of the item and based on the best offer continues with the order. Finally, when the items arrive they are added to the warehouse.

5.5.2.2 BPMN Model

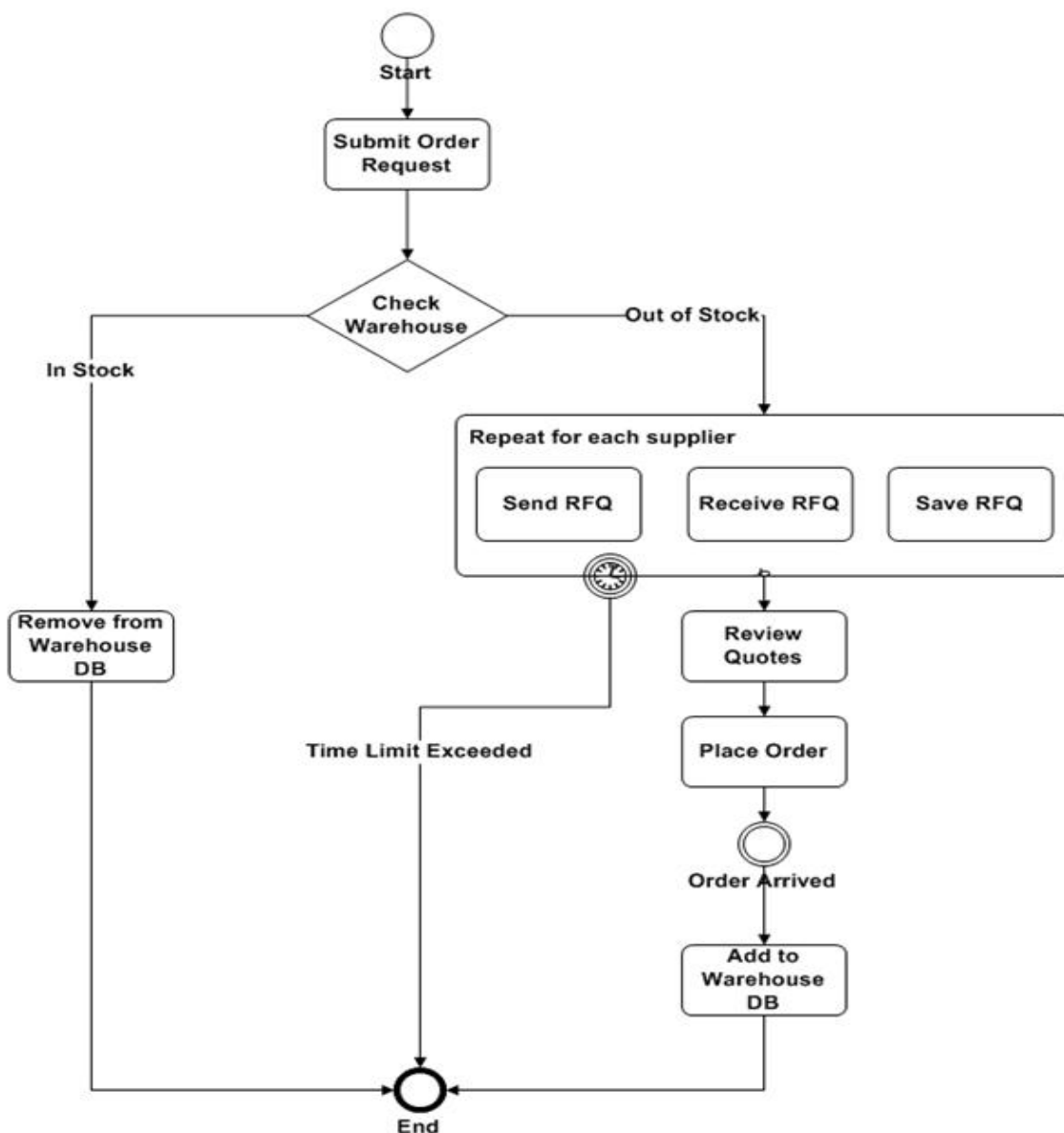


Figure 5.9: BPMN model: Supply Fulfilment Process [200]

5.5.3 Hiring Process

5.5.3.1 Process Description

This is a typical hiring process. Candidates submit job applications that are being reviewed by HR Managers. The applications are either approved at once, or totally disapproved, or more information is requested from the candidate. In all cases, the candidate is informed about the outcome of the review and in case he is asked for a revision, the candidate prepares a revised application form. Then the candidate resubmits the form and is informed about the outcome of the final review.

5.5.3.2 BPMN Model

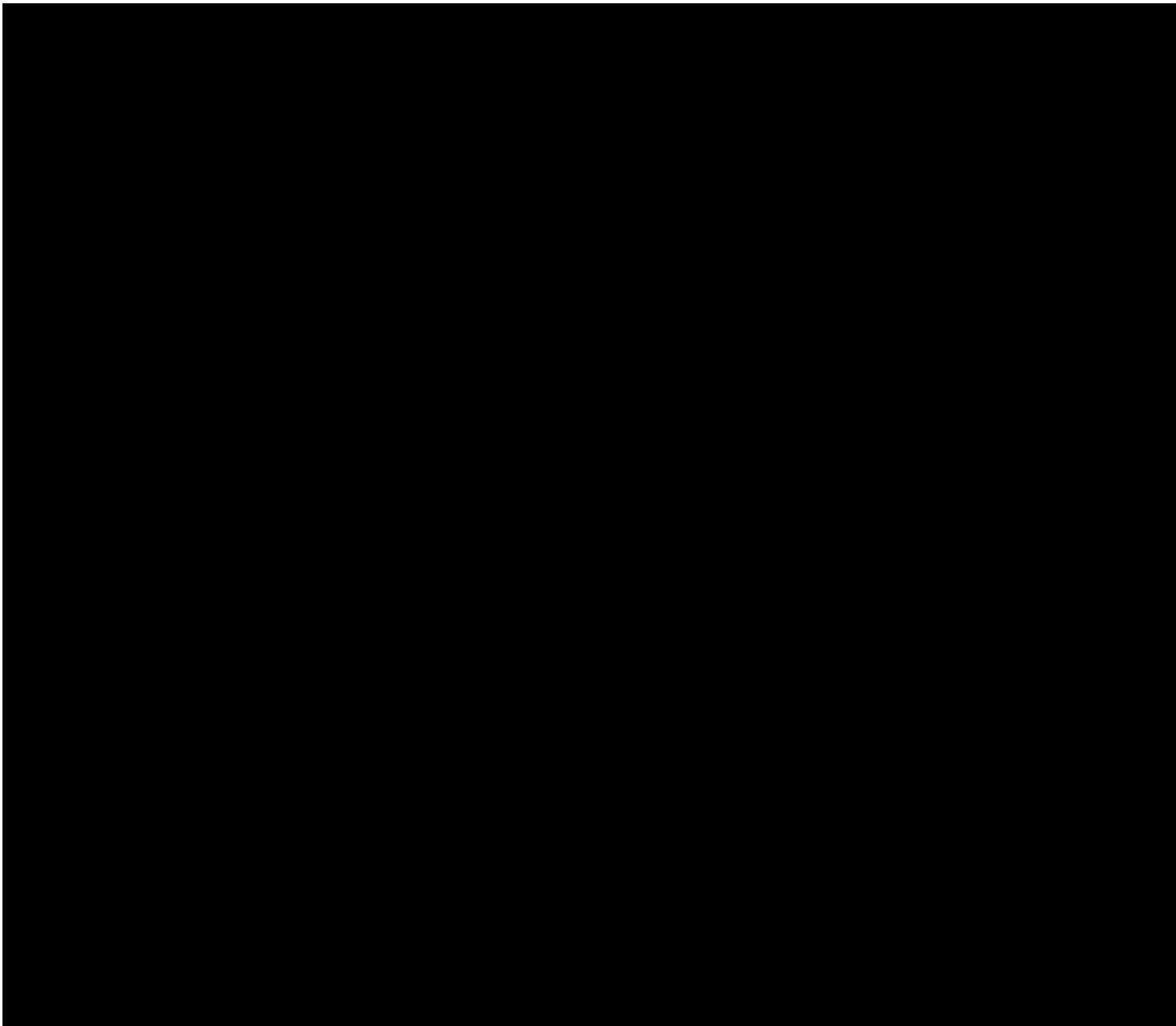


Figure 5.10: BPMN model: Hiring Process [201]

5.5.4 RFQ/Order/Payment collaboration Process

5.5.4.1 Process Description

“This model details three main components of the overall process: (1) The private process of the buyer, (2) the private process of the supplier and (3) collaboration between the buyer, the supplier, the bank and the credit check organization. The private process starts and as part of its "GetQuote" activity, it sends an RFQ document to the supplier. When the supplier receives this document, it dispatches it to its ProcessRFQ activity. This activity is invoking the PrepareQuote activity which is actually an activity performed by a user. As part of the activity, the user invokes the ManageAccount activity, which is in charge of creating a customer record if the customer is unknown and in all cases to perform a credit check with a partner agency. If all goes well, a quote is returned to the processRFQ activity, which forwards it to the customer. If the customer likes the quote, its process stipulates that it will start the SendOrder activity, which creates, and send an order relative to the quote. When the order is received by the supplier, it dispatches to the processRFQ activity, which ends at this point, and initiate the processOrder activity, which notifies the IssueInvoice activity, which sends an invoice as a response to the order. And so on.”

[202]

5.5.4.2 BPMN Model

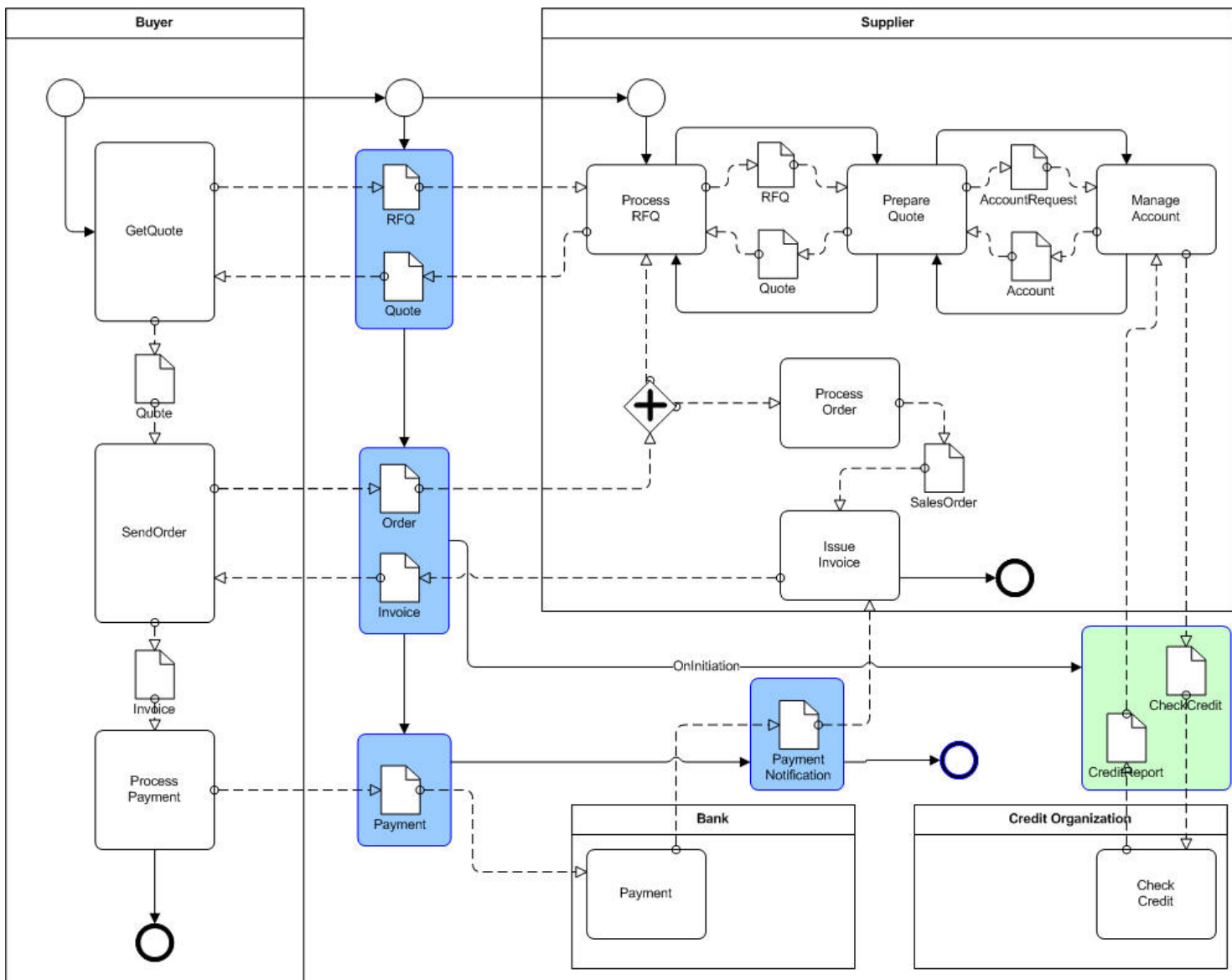


Figure 5.11: BPMN model: RFQ/Order/Payment collaboration Process [202]

5.5.5 Overall comparison

Process	BPMN	Flat SYMEX	Hierarchical SYMEX
Complaint Registration (Section 5.4)	0,0769	0,1581	0,1753
Trouble Ticket	0,0939	0,1931	0,2127
Supply Fulfillment	0,1282	0,2624	0,2873
Hiring Process	0,0578	0,1093	0,1214
Buy through credit	0,0623	0,1285	0,1437

Table 5.17: Calculation of CC-metric for all the cases

5.6 Conclusions

High complexity has undesirable effects on, among others, the validation, maintainability, and understandability of business process models. In this Chapter, we evaluated SYMEX framework with respect to the complexity of the produced process models. As, to the best of our knowledge, there is no other integrated framework with similar characteristics to SYMEX, we validated the framework against the closest similar research approach, which are the mappings we reviewed in Chapter 2. BPMN to BPEL4WS mapping is considered as the most significant and widely used out of all the mappings and therefore we used it as the basis for comparison with our approach.

In order to demonstrate that our approach is more efficient in producing easier to understand and maintain process models than the BPMN to BPEL4WS mapping, we have reviewed a number of complexity metrics for business process models. Most proposals of process metrics are inspired by software quality metrics due to similarities between software and processes. However, there are a number of metrics that are based to other domains and among them, we selected Cross-Connectivity (CC) metric as the basis for evaluating the SYMEX framework. CC is a validated and tested metric, based on cognitive science, which quantifies the complexity of a process model in terms of the cognitive effort of a model reader to understand it. CC metric has also an implementation independent of specific modelling approach and can be applied to both BPMN and SYMEX models.

The CC metric was adapted to take into account the special characteristics of SYMEX models. Then it was applied to a number of process models in BPMN, 'flat' SYMEX models and hierarchical SYMEX models. The results demonstrated that the hierarchical SYMEX models are less complex than both 'flat' SYMEX models and equivalent BPMN models. These findings confirm that SYMEX models are easier to understand and maintain than BPMN models and validate the benefit of reducing complexity by decomposing process models. However, in order to generalise these findings we need to extend this comparative evaluation approach with more modelling techniques.

6 Conclusions

In this final Chapter, we provide an overview of our work as presented in this Thesis, and then we identify the main contributions along with the limitations of our research. Finally, we present some directions for future research and conclude the Thesis with some final remarks.

6.1 Overview

Workflow management systems enable organisations to deal with all aspects of business process management, including analysis, modelling, execution, and administration. As workflow technology evolves, a landscape of languages and techniques for workflow process modelling has emerged and it is continuously being enriched with new proposals from different vendors and coalitions. Modelling workflow processes involves transformation of the process logic into a formal representation and it always remains a critical success factor for workflow management systems. However, common practice is using workflow modelling languages for capturing high-level descriptions of business processes. Then these descriptions are transformed by process designers into low-level execution semantics with the use of workflow programming languages. Thus, the two descriptions are maintained manually as completely different models.

Maintaining these models separately results in a number of issues. Inconsistency between the models is the primary issue. Specifically, certain information included in the high-level descriptions is either partly encoded or omitted from the low-level execution semantics and at the same time, complicated business rules encoded at the execution level are not included in the high-level descriptions. Moreover, maintaining these models separately creates difficulties in adaptation. Particularly, identifying changes in high-level descriptions due to modifications of business conditions, and tracing the impact of those changes on the low-level execution semantics can often become very difficult for business modellers and process designers. As a result, the performance and efficiency of workflow management systems is affected.

This thesis addresses the aforementioned problems by proposing a framework named SYMEX. SYMEX proposes a solution by integrating high and low-level descriptions in one unified format, from a Systems Theory perspective (Research Objective RO1). It provides a set of mathematically defined descriptions offering a rigid framework for capturing and representing both high-level business process descriptions and low-level workflow execution semantics. Furthermore, we have effectively tested the applicability of the framework, in terms of implementing a working prototype together with a workflow inference engine that executes SYMEX process descriptions (Research Objective RO2).

Finally, as high complexity results in undesirable effects on the validation, maintainability, and understandability of business process models, we have evaluated the produced process models

with respect to their complexity (Research Objective RO3). Adapting a process complexity metric with foundations on the cognitive domain (CC metric), we confirmed that SYMEX offers easy to understand and communicate set of constructs. The produced process models proved less complex than BPMN models. Additionally, hierarchy in SYMEX proved to reduce complexity of the models in comparison with 'flat' ones.

6.2 Summary of the Work

In **Chapter 1**, the focus of our research was set on the modelling aspect of business processes within a workflow management context. Then, we defined the thesis motivation as the problem of maintaining consistency between high-level representations and low-level execution process models. The three objectives of our research were defined together with the research methodology and the structure of the thesis.

In **Chapter 2**, we reviewed the literature with respect to workflow modelling languages used for the representation of high-level descriptions of business processes and workflow process execution frameworks used to define low-level execution semantics. We also reviewed some research approaches in the literature proposing mappings and discussed the enabling disciplines of our research: Systems Theory and Constrained-based reasoning. At the end of Chapter 2, we identified the limitations of the literature in respect with our research objectives.

In **Chapter 3**, we introduced our Systems Theory based framework for workflow modelling and execution, named SYMEX. The definitions of the basic concepts of SYMEX were followed by the formalization of its constructs. At the end of Chapter 3, we successfully assessed the framework's capability to effectively capture business processes in high-level descriptions as well as for encoding low-level execution semantics. In this Chapter, we have answered the following Research Questions:

- **RQ1.1:** How can we enable integration of high-level process descriptions and low-level workflow execution semantics in the SYMEX approach using the main principles of Systems Theory?

Answer: Inspired by Systems Theory we visualized a workflow model as an entity and the two models as the elements of this entity. In addition, the concept of hierarchy proposed by Systems Theory helped us to define hierarchical SYMEX models where the upper layers are depicting high-level descriptions and lower-layers encode the execution semantics.

- **RQ1.2:** What kind of conceptual constructs of Systems Theory can be used to model high-level and low-level workflow process descriptions?

Answer: Modelling the workflow process from a Systems Theory perspective, SYMEX models view workflow processes as a set of Activities, which take Inputs and produce Outputs. Activities are executed via the means of their execution Mechanisms and activated by their Control, which evaluates a number of constraints.

- **RQ1.3:** How can we formally describe systems oriented process models that encode both high-level and low-level workflow process descriptions?

Answer: Having provided an informal definition of the main concepts of SYMEX, we have defined the mathematical descriptions of constructs and their relationships based on set theory, which allows reasoning about the process characteristics at different levels of detail, in a more controlled way.

In **Chapter 4**, we proposed an XML-based implementation approach in order to create a working prototype of SYMEX. Then we presented an example of capturing a business process using SYMEX modelling and tested our framework's adaptation in comparison with UML 2 Activity Diagrams. At the end of Chapter 4, we presented the constraint-based algorithm of an implemented workflow inference engine that executes SYMEX models. In this Chapter, we have answered the following Research Questions:

- **RQ2.1:** How can systems oriented process descriptions be defined in order to model real life business workflows?

Answer: We have encoded the constructs and relationships of SYMEX using XML technology, in a top-down approach. We ended with a Schema Definition (XSD) that enables the definition of SYMEX models in XML format that can be easily processed by computers.

- **RQ2.2:** What is required in order to execute systems oriented business workflows?

Answer: Based on constraint-based reasoning, which is a problem-solving approach, we conceptualized the problem of workflow execution in terms of a constraint satisfaction problem. Then we described the constraint-based algorithm we have implemented for our workflow inference engine, which is capable of executing SYMEX workflow models.

Finally, in **Chapter 5** we evaluated SYMEX framework with respect to the complexity of the produced process models, in comparison with BPMN models. In this Chapter, we have answered the following Research Question:

- **RQ3:** What kind of measures can we define in order to assess effectiveness; how easy is to understand and manage changes in SYMEX models, compared to existing available techniques?

Answer: In order to prove that SYMEX is more efficient in producing easier to understand and maintain process models, we selected and adapted the Cross-Connectivity metric. With the use of this metric, we quantified the complexity of the produced models. The results validated that SYMEX models are easier to understand and maintain than BPMN models and validated the benefit of complexity reduction in SYMEX process models with decomposition.

6.3 Contributions

The contributions of this thesis are the results of our work to address the research objectives.

6.3.1 Integration of high-level process descriptions and low-level workflow execution semantics into SYMEX

One of the most important contributions of our work is the integration of high-level process descriptions and low-level execution semantics into one unified format. Up to now, there are modelling languages that capture business processes and create visual high-level process models on one hand and there are execution frameworks, which are used to define the low-level execution semantics for the processes that are eventually executed by an execution engine, on the other hand. Those two descriptions are mostly maintained and adapted manually, even though there are some proposals for mappings, trying to reduce the gap between them. This common practice creates difficulties and results in high maintenance costs. Therefore, the problem of finding an efficient way to integrate high-level process descriptions and low-level execution semantics was our primary research challenge.

In order to address the issue of integration, we employed Systems Theory as an enabling discipline. Systems Theory defines a system as a set of elements standing in interrelation among them and with environment. The Theory focuses on the arrangement of and relations between the parts or elements of an entity and connects them into a whole. Moreover, Systems Theory introduces hierarchy for determining multiple layers with different detail of description. Inspired by these concepts, we considered workflow model as an entity, while high-level description and low-level executable description were considered as two elements of this entity. Therefore, the problem of integration became a problem of defining the arrangement of and the relations between these two elements and their parts. The solution was to conceptualize the high-level description as the upper layer element and the low-level executable description as the lower layer element of a hierarchical decomposition of the 'entity' workflow model. Both elements were also connected into a whole, inside the 'entity' workflow model.

The result of thinking by concepts of Systems Theory was the definition of SYMEX, an integrated framework for workflow modelling and execution, with simple and basic concepts that are easy to learn and understand. Specifically, SYMEX models are defined as a hierarchical set of Activities having controls and execution mechanisms and being interconnected with inputs and outputs. Composite Activities are further decomposed to more detailed layers, until atomic Activities are reached, which are essentially the execution parts of the process model. Thus, the high-level description of a workflow process is modelled by composite Activities in higher layers, while the low-level execution semantics are modelled by atomic Activities in lower layers of the SYMEX model. SYMEX models can also enhance communication between designers and programmers, as both of

them work at different layers of the same model, avoiding manual transformations from one description to another. Finally, we have formally defined the semantics of SYMEX modelling and validated its capability to efficiently represent business dynamics as well as encode workflow execution patterns.

To the best of our knowledge, there is no similar research approach for the problem of maintaining separate models of high-level process descriptions and low-level execution semantics, classifying the result of this work as a pioneering contribution in the workflow modelling research area.

6.3.2 Constraint-based process modelling and execution

Another contribution of our work is the use of constraint-based reasoning for workflow process modelling. Common practice in modelling a business process is the construction of linear sequences of activities. However, this linear approach creates problems and often does not work because the business world is not linear. Although, some researchers have incorporated constraints with business process modelling, these research efforts remain theories and all of the reviewed modelling languages create models of business processes by sequencing the order of their activities. Constraint-based reasoning enabled us to effectively model non-linear relationships between activities in SYMEX models. In particular, constraints are encoded as controls in each Activity of the model and there is no explicit sequence. In this way, Activities in SYMEX models are not dependent with each other in terms of explicit control flow constructs.

The lack of execution dependencies of Activities in SYMEX models helps us address changing conditions within business processes; it is much easier to adapt a workflow process that is constrained, by adding or changing the constraints of its Activities, than to adapt a sequenced workflow process. When a process is sequenced using traditional programming constructs (if-then-else, cases, while-loops etc.), the designer needs to resolve all the sequence of activities and the dependencies between them in order to adapt the process to new business needs. On the other hand, in SYMEX models the process designer focus only on the Activity that needs to be changed, not having to interfere with any other irrelevant Activities. Thus, the constraint-based approach of SYMEX modelling enables easy and quick adaptation of process models in response to business change.

6.3.3 Application of SYMEX

An additional result of our work was the successful assessment of the applicability of SYMEX. Specifically, we have proposed an implementation of SYMEX that enables us to capture and execute integrated workflow process models. The implementation is based on XML and is essentially a Schema Definition (XSD), which encodes the concepts, properties, and relationships of SYMEX modelling in a top-down approach. This essentially provides a persisted model of process descrip-

tions modelled with SYMEX, which then can be easily executed by computers and could be easily supported by graphical tools.

Together with the proposed XML-based implementation of our framework, we presented the implementation approach of a workflow inference engine that executes SYMEX models. The algorithm of the engine is based on constraint-based theory and the execution of SYMEX models is a solution of a constraint-satisfaction problem; each Activity has a number of constraints that need to be satisfied in order to execute. Moreover, the implemented algorithm is not complex and can be easily implemented with any programming language. The first version of the engine has been implemented using MS Visual Basic 6, but the code could be easily immigrated to Java [186] or .Net platform.

6.3.4 Adaptation of CC metric and evaluation of SYMEX models

A final contribution of our work is the adaptation of the Cross-Connectivity (CC) metric and the quantification of SYMEX modelling benefits. In order to evaluate the SYMEX models in comparison with BPMN models, we selected the CC metric, which quantifies the complexity of a process model in terms of the cognitive effort of a model reader to understand it. We adapted the formulas of CC metric suitably to take into account special characteristics of SYMEX models. The CC metric was then applied to a process model in BPMN, a 'flat' SYMEX model and a hierarchical SYMEX model. The results demonstrated that hierarchical SYMEX model was less complex than both 'flat' SYMEX model and the equivalent BPMN model. Thus, the findings confirmed that SYMEX models are easier to understand and maintain than BPMN models and that decomposition reduces their complexity.

6.3.5 Publications

Our research produced a number of publications listed below in chronological order:

- Bill Karakostas, Yannis Zorgios & Charalampos C. Alevizos, Automatic derivation of BPEL4WS from IDEF0 process models. *Journal of Software & System Modelling*, Springer Berlin / Heidelberg, 2006. 5(2): p. 208-218.
- Bill Karakostas, Yannis Zorgios & Charalampos C. Alevizos, The Semantics of Business Service Orchestration, *Advances in Semantics for Web services 2006 Workshop (semantics4ws'06)*, 2006, Vienna, Austria.
- Charalampos C. Alevizos, Bill Karakostas & Yannis Zorgios, Complexity Calculation of service orchestration models, submitted to 4th International Conference on Software and Data Technologies - ICSOFT 2009, Sofia, Bulgaria.

6.3.6 POMPEI Project

The constraint-based process modelling approach of SYMEX was used in an initial form in POMPEI [203]. POMPEI was European-funded project, under the Sixth Framework Programme (FP6), that aimed to develop a dynamic, mobile, and peer-to-peer workflow management system for emergencies. Our constraint-based approach was used as the workflow layer, responsible for the execution of processes that were modelled in terms of Activities, Inputs, Outputs, Controls, and Mechanisms. The implementation was based on Java's [186] edition for mobile devices, namely J2ME.

6.4 Research Limitations

The limitations we identify in our research are:

- The control flow in SYMEX models is encoded inside Controls of Activities and is not visually represented. As a result, when a user reads a SYMEX model he cannot understand the sequence of Activities but only the information flow between them (via Inputs and Outputs). This limitation is due to the constraint-based approach we used for SYMEX modelling and is the basic trade-off for easy and quick adaptation.
- In case of modelling extremely large business processes, the SYMEX model may consist of too many decomposition levels that it would be very difficult for a user to follow and understand it.
- The proposed implementation approach for SYMEX is based on XML. XML may be open and standard but we consider it also as a strict and rudimentary language. It could turn as a disadvantage; in case we would try to enrich our process descriptions with more semantics (such as OWL-s) in order to be able to support automatic discovery and execution of services.
- The implemented workflow inference engine for SYMEX models has not been thoroughly tested. Besides this, it does not support web-service execution.
- The evaluation of SYMEX models, in terms of their complexity, was done in comparison with BPMN models. In order to generalize the results and validate the benefits of SYMEX modelling, more comparative evaluation with existing workflow modelling languages and execution frameworks needs to be done.
- There is no tool available yet for SYMEX modelling.
- Finally, although SYMEX modelling may be easy to learn and straightforward, the fact is that most designers and developers use and trust well-established modelling approaches, such as UML Activity Diagrams and BPMN, and are reluctant to try something new and not extensively tested.

6.5 Future Work

This thesis introduced SYMEX as an integrated framework for workflow modelling and execution. Whatever the benefits of our proposal may be, without a graphical tool that would enable analysts, designers and end users to create, modify and communicate SYMEX models, we could not expect quick adoption from academia and industry. For this reason, we are in the early stages of developing a graphical tool that has an easy to use interface with drag and drop features, in order to make the creation of SYMEX models easy and accessible to everyone. We have not discussed earlier in the thesis about this tool because it is in a very premature developing phase and needs a lot of testing and development before it can be considered even a beta version.

As far as the execution capabilities of SYMEX are concerned, we have reviewed in Sections 3.4 and 3.5 of Chapter 3 the support of workflow patterns. There are three workflow patterns that cannot be supported by SYMEX and one that is partially supported. One of our concerns is whether we could extend SYMEX framework in order to offer complete support of workflow execution patterns. Some early attempts did not have any success and the question that we will have to answer is what kind of changes have to be made and what will be the cost of these changes in terms of adding new constructs and making SYMEX approach more complex to use and maintain.

Moreover, Section 4.5 introduced the implemented workflow inference engine that executes processes modelled using SYMEX. Current implementation supports the execution of components installed in the same computer with the installed engine. However, the main objective of workflow technology is ubiquitous distributed execution of business processes across organizational and platform boundaries. Web-services has come as platform independent software services in order to enable this objective and one of our main goals is to extend the workflow inference engine in order to support synchronization and execution of web-services across Internet.

As for SYMEX's evaluation, in Chapter 5, we used the Cross-Connectivity metric in order to evaluate the efficiency in terms of producing easier to understand and maintain SYMEX process models than BPMN models. In order to prove that SYMEX models are generally easier to adapt and maintain than other business process models, we need to extend the comparative evaluation with more modelling techniques, beginning with those reviewed in Chapter 2; IDEF0, EPC, IDEF4, XPDL, BPDM and UML2 ADs. Moreover, we need to compare SYMEX to process execution frameworks as well. The evaluation at this level will include the capability of supporting workflow patterns and the capability of working with existing workflow engines. BPEL4WS, OWL-S and WSMX will be the first candidates for comparison but we will also consider other industrial/academic proposals for workflow execution.

Overall, industries and organizations today make significant investments, in time and money, on the modelling approaches they follow. Usually, when there is a new and more efficient modelling proposal, there is the dilemma of either continuing with the existing modelling approach with all the disadvantages it may have or choosing the new proposal with all the promising features and capabilities, but also with all the hassle and the cost of the incurred change. The answer to this dilemma is to follow the Java paradigm. We intend to examine whether SYMEX framework can be adapted in order to play the role of a neutral workflow modelling approach, which would enable us to apply the Java paradigm to the area of workflow process modelling; be able to model a workflow process only once in SYMEX and execute it using any execution framework and engine, such as BPEL4WS, WSMX etc.

6.6 Final Remarks

Software engineering is changing and software design is moving from monolithic structures to multi-layer distributed architectures. Workflow technology plays a key role in the new era of application architectures. Initially when workflow technology was introduced, it was employed to address document routing and gradually was adopted inside software systems. The first automated business applications had all workflow related information hard-coded inside the software components. With the evolution of software development, workflow has become an independent layer (the workflow layer).

Workflow technology includes aspects of modelling, execution and administration of business processes. Current modelling and execution approaches and methods, such as BPMN, UML 2 ADs, BPEL4WS etc. focus either on the modelling phase or on the execution phase. These two different phases are maintained as two separate models resulting in synchronization and maintenance issues. What we have proposed in this thesis is an integrated view of modelling and execution semantics of workflow process. Given the lack of previous research results in this area, we adopted a systems perspective and visualized a workflow model as a hierarchical model where the upper layers are depicting high-level descriptions and lower layers encode the execution semantics. To the best of our knowledge, there is no equivalent proposal from the industry or the academia following our systems oriented approach.

Our research was guided by the vision to define SYMEX as a methodology and a platform independent workflow modelling and execution language in order to play the same role as Java plays in software development. Java has become a language where programmers use to develop their programs and then these programs can run on any available platform (Windows, Linux, Solaris etc.) that supports the Java Virtual Machine. So Java promotes the idea "program once and run everywhere". In the same sense, SYMEX could play the role of a platform-independent modelling approach where process designers would model the processes once and then these models could be run in any execution language/engine, e.g. a process would be modelled in SYMEX and then could be executed either in our workflow inference engine or in a BPEL engine.

The area of workflow management is continuously changing, shaped by new paradigms as Service-Oriented Architecture (SOA) [204] and ubiquitous distributed execution across organizational and platform boundaries. Following the recent trends, workflow technology is adopting the use of web-services [21], as an infrastructure providing seamless interoperability among networked workflow processes [22]. The presence of web-services has raised issues of monitoring and controlling their execution within the workflow context they operate. However, the importance of workflow process modelling remains a semantic issue, facing new sort of challenges.

In conclusion, we believe that this thesis made a definitive contribution to the quest for more flexible and dynamic business process systems, by providing a new perspective for conceptualizing workflow process modelling, that we hope can serve as a constructive basis for further research on this area.

References

- [1] C. Badica and C. Fox, "Modelling and Verification of Business Processes," in *IASTED International Conference Applied Simulation and Modelling (ASM 2002)*, Crete, Greece, 2002.
- [2] B. List and B. Korherr, "An evaluation of conceptual business process modelling languages," in *ACM symposium on Applied computing* Dijon, France: ACM Press, 2006
- [3] R. Endl, G. Knolmayer, and et al., "Modeling Processes and Workflows by Business Rules," Institute of Information Systems, University of Bern 1998.
- [4] A. Lonjon, "Business Process Modeling and Standardization," BPTrends 2004.
- [5] T. Davenport, *Process Innovation: Reengineering work through information technology*. Harvard Business School Press, Boston, 1993.
- [6] "Business Processes and the OMG: An overview," Object Management Group, <http://www.omg.org>.
- [7] "Workflow Management Coalition," www.wfmc.org.
- [8] L. Fischer, "An Introduction to Workflow," in *Workflow Handbook 2002*: Workflow Management Coalition, 2002.
- [9] M.-T. Schmidt, "The Evolution of Workflow Standards," *IEEE Concurrency*, pp. 44-52, 1999.
- [10] D. Hollingsworth, "Workflow Management Coalition: The Workflow Reference Model," The Workflow Management Coalition 1995.
- [11] S. Shazia, S. Olivera, M. Maria, and E. Orlowska, "Managing Change And Time In Dynamic Workflow Processes," *International Journal of Cooperative Information Systems*, 1999.
- [12] W. M. P. v. d. Aalst, "Generic Workflow Models: How to Handle Dynamic Change and Capture Management Information?," in *Fourth IECIS International Conference on Cooperative Information Systems*, Edinburgh, Scotland, 1999, p. p. 115.
- [13] L. Zeng, A. Ngu, B. Benatallah, and M. O'Dell, "An Agent-Based Approach for Supporting Cross-Enterprise Workflows," in *Australasian Database Conference*, Gold Coast, Queensland, Australia, 2001, p. p. 0123.
- [14] A. I. Wang, "Using software agents to support evolution of distributed workflow models," in *International ICSC Symposium on INTERACTIVE AND COLLABORATIVE COMPUTING (ICC'2000)*, Wollongong (near Sydney), Australia, 2000.
- [15] "Common Object Request Broker Architecture (CORBA®)," Object Management Group <http://www.corba.org/>.
- [16] M. Purvis, M. Purvis, and S. Lemalu, "A Framework for Distributed Workflow Systems," in *34th Annual Hawaii International Conference on System Sciences (HICSS-34)*, Maui, Hawaii, 2001, p. p. 9039.
- [17] S. Paul, E. Park, and J. Chaar, "RainMan: A Workflow System for the Internet," in *USENIX Symposium on Internet Technologies and Systems*, 1997.
- [18] J. G. Hayes, E. Peyrovian, S. Sarin, M.-T. Schmidt, K. D. Swenson, and R. Weber, "Workflow Interoperability Standards for the Internet," *IEEE Internet Computing*, 2000.
- [19] G. A. Bolcer, "Magi: An Architecture for Mobile and Disconnected Workflow," *IEEE Internet Computing*, pp. 46-54, 2000.
- [20] R. Tolksdorf, "Workspaces: A Web-Based Workflow Management System," *IEEE Internet Computing*, 2002.
- [21] M. Virdell, "Business processes and workflow in the Web services world," IBM developer-Works 2003.
- [22] F. Casati, E. Shan, U. Dayal, and M.-C. Shan, "Business-oriented management of Web services," *Communications of the ACM*, 2003.
- [23] W3C-WSDL, "Web Services Description Language (WSDL), <http://www.w3.org/TR/wsdl>," 2001.
- [24] W3C, "SOAP Version 1.2 <http://web5.w3.org/TR/2003/REC-soap12-part0-20030624/>," XML Protocol Working Group, 2003.
- [25] F. Cabrera, G. Copeland, B. Cox, T. Freund, J. Klein, T. Storey, and S. Thatte, "Web Services Transaction (WS-Transaction)," 2002.

-
- [26] S. Overhage and P. Thomas, "WS-Specification: Specifying Web Services Using UDDI Improvements," in *Web, Web Services, and Database Systems - NODe 2002 Web- and Database-Related Workshops*, 2002.
 - [27] F. Cabrera, G. Copeland, T. Freund, J. Klein, D. Langworthy, D. Orchard, J. Shewchuk, and T. Storey, "Web Services Coordination (WS-Coordination)," 2002.
 - [28] A. Bosworth, D. Box, E. Christensen, F. Curbera, D. Ferguson, J. Frey, C. Kaler, D. Langworthy, F. Leymann, S. Lucco, S. Millet, N. Mukhi, M. Nottingham, D. Orchard, J. Shewchuk, T. Storey, and S. Weerawarana, "Web Services Addressing (WS-Addressing)," 2003.
 - [29] L. Ardissono, A. Goy, and G. Petrone, "Enabling conversations with Web Services," in *2nd Int. Conf. on Autonomous Agents and Multi Agent System (AAMAS'03)*, Melbourne, Australia, 2003.
 - [30] K. Matsumura, H. Mizutani, and M. Arai, "An Application of Structural Modeling to Software Requirements Analysis and Design," *IEEE Transactions on Software Engineering*, 1987.
 - [31] *Systems Engineering Fundamentals*. Virginia, USA: Defense Acquisition University Press, 2001.
 - [32] A. Abran, J. W. Moore, P. Bourque, and R. Dupuis, *Guide to the software engineering body of knowledge*: IEEE Computer Society Press, 2005.
 - [33] "UML 2 Activity Diagram," Sparx Systems Pty Ltd. - http://www.sparxsystems.com.au/resources/uml2_tutorial/uml2_activitydiagram.html.
 - [34] "Business Process Execution Language for Web Services (BPEL4WS)," BEA, IBM, Microsoft, 2003.
 - [35] G. Keller, M. Nüttgens, and A.-W. Scheer, "Semantische Prozeßmodellierung auf der Grundlage "Ereignisgesteuerter Prozeßketten (EPK)", " Veröffentlichungen des Instituts für Wirtschaftsinformatik, Saarbrücken 1992.
 - [36] "INTEGRATION DEFINITION FOR FUNCTION MODELING (IDEF0)," <http://www.idef.com/idef0.html>.
 - [37] P. D. Richard J. Mayer, P. D. Christopher P. Menzel, M. K. Painter, P. D. Paula S. deWitte, T. Blinn, and P. D. Benjamin Perakath, "INFORMATION INTEGRATION FOR CONCURRENT ENGINEERING (IICE) IDEF3 PROCESS DESCRIPTION CAPTURE METHOD REPORT," KNOWLEDGE BASED SYSTEMS, INCORPORATED 1995.
 - [38] "Business Process Modeling Notation," Business Process Management Initiative, <http://www.bpmn.org/> 2002.
 - [39] "Workflow Process Definition Interface – XML Process Definition Language (XPDL), Document Number WPMC-TC-1025, Version 1.0,," Workflow Management Coalition, 2002.
 - [40] OMG, "UML 2.0 superstructure specification, Technical report," 2004.
 - [41] "Web Services Flow Language (WSFL)," <http://www4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf>: IBM, 2001.
 - [42] "XLANG," http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm: Microsoft, 2001.
 - [43] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana, "Business Process Execution Language for Web Services Version 1.1," BEA Systems, International Business Machines Corporation, Microsoft Corporation, SAP AG, Siebel Systems, 2003.
 - [44] "Web Ontology Language (OWL)," <http://www.w3.org/2004/OWL/>: W3C Recommendation.
 - [45] "Web Service eXecution Environment (WSMX)," WSMX working group.
 - [46] K. Mantell, "From UML to BPEL," SOA and Web Services, developerWorks, IBM, 2003.
 - [47] T. Gardner, "UML Modelling of Automated BusinessProcesses with a Mapping to BPEL4WS," in *17th European Conference on Object-Oriented Programming (ECOOP)*, Darmstadt, Germany, 2003.
 - [48] T. Gardner, "UML Modelling of Automated Business Processes with a Mapping to BPEL4WS," in *First European Workshop on Web Services and Object Orientation, ECOOP 2003*: IBM UK Laboratories, Hursley Park, 2003.

-
- [49] C. Ouyang, W. M. P. v. d. Aalst, M. Dumas, and A. H. M. t. Hofstede, "From BPMN Process Models to BPEL Web Services," in *4th International Conference on Web Services (ICWS)*, Chicago IL, USA, 2006.
 - [50] C. Ouyang, W. M. P. v. d. Aalst, M. Dumas, S. Breutel, and A. H. M. t. Hofstede, "Translating BPMN to BPEL," BPM Report BPM-06-02, BPMcenter.org 2006.
 - [51] C. Ouyang, W. M. P. v. d. Aalst, M. Dumas, S. Breutel, and A. H. M. t. Hofstede, "From business process models to process-oriented software systems: The BPMN to BPEL way," BPM Center Report BPM-06-27, BPMcenter.org 2006.
 - [52] S. A. White, "Introduction to BPMN," IBM Corporation 2004.
 - [53] M. G. Nadarajan and D. Y.-H. Chen-Burger, "An Ontology-Based Conceptual Mapping Framework for Translating FBPML to the Web Services Ontology," in *Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2006)* Singapore, 2006.
 - [54] D. S. Frankel, *Model Driven Architecture: Applying MDA to Enterprise Computing*. John Wiley & Sons, 2003.
 - [55] "Object Management Group, <http://www.omg.org/>."
 - [56] K. Czarnecki and S. Helsen, "Classification of Model Transformation Approaches," in *OOP-SLA'03 Workshop on Generative Techniques in the Context of Model-Driven Architecture*, 2003.
 - [57] "MOF 2.0 Query / Views / Transformations RFP," Object Management Group 2002.
 - [58] "Jamda: The Java Model Driven Architecture 0.2, <http://sourceforge.net/projects/jamda/>," 2003.
 - [59] "Java Metadata Interface 1.0, <http://java.sun.com/products/jmi/>," 2002.
 - [60] D. H. Akehurst and S. Kent., "A Relational Approach to Defining Transformations in a Metamodel," in *The Unified Modeling Language 5th International Conference*, Dresden, Germany, 2002.
 - [61] "QVT-Partners. MOF Query/Views/Transformations, Revised Submission.," OMG Document: ad/2003-08-08 2003.
 - [62] CBOP, DSTC, and IBM, "MOF Query/Views/Transformations, Revised Submission.," OMG Document: ad/03-08-03 2003.
 - [63] A. Gerber, M. Lawley, K. Raymond, J. Steel, and A. Wood, "Transformation: The Missing Link of MDA," in *Graph Transformation: First International Conference (ICGT 2002)*, Barcelona, Spain, 2002.
 - [64] Alcatel, Softeam, Thales, TNI-Valiosys, and C. Corporation, "MOF Query/Views/Transformations, Revised Submission," OMG Document: ad/03-08-05 2003.
 - [65] M. Andries, G. Engels, A. Habel, B. Hoffmann, H.-J. Kreowski, S. Kuske, D. Kuske, D. Plump, A. Schürr, and G. Taentzer, "Graph Transformation for Specification and Programming," Universität Bremen 1996.
 - [66] D. Varro, G. Varro, and A. P. . "Designing the automatic transformation of visual languages," *Science of Computer Programming*, vol. 44, pp. 205-227, 2002.
 - [67] "ATOM3: A Tool for Multi-Paradigm modeling," <http://atom3.cs.mcgill.ca/>.
 - [68] A. Agrawal, G. Karsai, and F. Shi, "Graph Transformations on Domain-Specific Models," *Software and Systems Modeling*, 2003.
 - [69] E. D. Willink, "UMLX: A graphical transformation language for MDA," in *Workshop on Model Driven Architecture: Foundations and Applications*, University of Twente, Enschede, The Netherlands, 2003.
 - [70] P. Braun and F. Marschall, "The Bi-directional Object-Oriented Transformation Language," Technische Universität München 2003.
 - [71] I. Objects and P. Technology, "MOF Query/Views/Transformations, Revised Submission," OMG Document: ad/03-08-11, ad/03-08-12, ad/03-08-13 2003.
 - [72] Alcatel, Softeam, Thales, TNI-Valiosys, and C. Corporation, "MOF Query/Views/Transformations, Revised Submission," OMG Document: ad/03-08-05.
 - [73] J. Bézivin, G. Dupé, F. Jouault, and J. E. Rougui, "First experiments with the ATL model transformation language: Transforming XSLT into XQuery," in *Workshop on Generative Techniques in the Context of the MDA*, 2003.
 - [74] "Business Process Management Initiative, BPMI," <http://www.bpmi.org/>: W3C, OASIS, OMG.

-
- [75] "Organization for the Advancement of Structured Information Standards, OASIS, <http://www.oasis-open.org/>."
 - [76] "National Institute of Standards and Technology," www.nist.gov.
 - [77] W. M. P. v. d. Aalst and A. H. M. t. Hofstede, "YAWL: Yet Another Workflow Language," *Information Systems*, vol. 30, pp. 245-275, 2003.
 - [78] M. Vasko and S. Dustdar, "A view based analysis of workflow modeling languages," in *14th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP'06)*, 2006.
 - [79] G. M. GIAGLIS, "A Taxonomy of Business Process Modeling and Information Systems Modeling Techniques," *The International Journal of Flexible Manufacturing Systems*, 2001.
 - [80] P. Wohed, W. M. P. v. d. Aalst, M. Dumas, and A. H. M. t. Hofstede, "Analysis of Web Services Composition Languages: The Case of BPEL4WS " in *22nd International Conference on Conceptual Modeling (ER)* Chicago, IL, USA: Springer Verlag, 2003.
 - [81] S. Carlsen, J. Krogstie, A. Sølvsberg, and O. I. Lindland, "Evaluating Flexible Workflow Systems," in *Hawaii International Conference on System Sciences (HICSS-30)* Maui, Hawaii, 1997.
 - [82] "Business Process Definition Metamodel," Object Management Group, <http://www.bpmn.org/Documents/BPDM/OMG-BPD-2004-01-12-Revision.pdf>, 2004.
 - [83] "IEEE Computer Society," www.computer.org.
 - [84] "American National Standards Institute - ANSI," www.ansi.org.
 - [85] "IEEE Standard for Functional Modeling Language—Syntax and Semantics for IDEF0 [IEEE Std 1320.1-1998]," Software Engineering Standards Committee of the IEEE Computer Society 1998.
 - [86] "Draft Federal Information Processing Standards Publication 183, Standard for integration definition for function Modeling (IDEF0)," National Institute of Standards and Technology (NIST), 1993.
 - [87] P. Bernus, K. Mertins, and G. Schmidt, *Handbook on Architectures of Information Systems: The IDEF Family of Languages*. Springer Berlin Heidelberg, 1998.
 - [88] A. Plaia and A. Carrie, "Application and assessment of IDEF3-process flow description capture method," *International Journal of Operations & Production Management*, 1995.
 - [89] A.-W. Scheer, *Architecture of Integrated Information Systems: Foundations of Enterprise Modelling*. Springer-Verlag New York, Inc., 1992.
 - [90] Ferdian, "A Comparison of Event-driven Process Chains and {UML} Activity Diagram for Denoting Business Processes," in *Information and Communication Systems*. Technische Universität Hamburg-Harburg, 2001.
 - [91] Bezivin, A. Tsalgatidou, F. Vermaut, L. Kutvonen, and P. F. Linington, "State-of-the art for Interoperability architecture approaches," Network of Excellence - www.interop-noe.org 2005.
 - [92] A. Tsai, J. Wang, W. Tepfenhart, and D. Rosea, "EPC Workflow Model to WIFA Model Conversion," in *IEEE International Conference on Systems, Man and Cybernetics*, 2006.
 - [93] "The Event-driven Process Chain," in *ARIS Design Platform* London: Springer 2007.
 - [94] W. M. P. v. d. Aalst, "Formalization and Verification of Event-driven Process Chains," *Information and Software Technology*, 1999.
 - [95] W. M. P. v. d. Aalst, "Formalization and Verification of Event-driven Process Chains," Eindhoven University of Technology, Eindhoven 1998.
 - [96] A. Tsalgatidou, "Business Process Models," Department of Informatics and Telecommunications University of Athens, Athens.
 - [97] I. Knowledge Based Systems, "IDEF3 Process Description Capture Method," Knowledge Based Systems, Inc.
 - [98] Y.-H. Chen-Burger, A. Tate, and D. Robertson, "Enterprise Modelling: A Declarative Approach for FBPM," in *European Conference of Artificial Intelligence, Knowledge Management and Organisational Memories Workshop*, Lyon, France, 2002.
 - [99] J. Lee, H. Sarjoughian, F. Simcox, S. Vahie, and B. Zeigler, "A Group-Based Approach for Distributed Model Construction," in *HICSS - Proceedings of the Thirty-First Annual Hawaii International Conference on System Sciences*, 1998, p. 220.

- [100] J. Eatock, R. J. Paul, and A. Serrano, "A Study of the Impact of Information Technology on Business Processes Using Discrete Event Simulation: A Reprise," *International Journal of Simulation Systems, Science & Technology*, vol. 2, pp. 30-40, 2001.
- [101] "Business Process Modeling Notation, OMG Available Specification," Object Management Group 2008.
- [102] S. A. White, "Using BPMN to Model a BPEL Process," IBM Corp 2005.
- [103] T. Crusson, "Business Process Management Essentials," GLINTECH 2006.
- [104] W. M. P. v. d. Aalst, A. H. M. t. Hofstede, B. Kiepuszewski, and A. P. Barros., "Workflow Patterns," Queensland University of Technology, Brisbane 2002.
- [105] A. Cicortas, K.-P. Eckert, F. Fortis, B. Gaillard, Y. Glickman, J. Hall, R. Knapik, and R. Renk, "VISP Workflow Technologies Functional Analysis and Comparison," Information Society Technologies 2006.
- [106] "XPDL," Workflow Managemnt Coalition.
- [107] W. M. P. v. d. Aalst, "Patterns and XPDL: A Critical Evaluation of the XML Process Definition Language," Queensland University of Technology, Brisbane 2003.
- [108] "WfMC Sample Workflow," The Open Business Engine.
- [109] "Altova XML Spy 2007," ALTOVA - <http://www.altova.com/>.
- [110] J.-J. Dubray, "XPDL Analysis," ebPML.org, 2002.
- [111] H. Salah, B. Mahmoud, and B. Nacer, "An architecture for the interoperability of workflow models," in *first international workshop on Interoperability of heterogeneous information systems*, Bremen, Germany, 2005.
- [112] "Business Process Definition MetaModel," Object Managemet Group 2008.
- [113] "Business Process Definition Metamodel - Request For Proposal," Object Management Group 2003.
- [114] "XML Metadata Interchange (XMI)," Object Management Group - <http://www.omg.org/technology/documents/formal/xmi.htm> 2007.
- [115] A. Chazalet and P. Lalande, "A Meta-Model Approach for the Deployment of Services-oriented Applications," in *IEEE International Conference on Services Computing*, 2007.
- [116] OMG, "Meta Object Facility."
- [117] modeldriven.org, "Business Process Definition Metamodel."
- [118] R. Gronback, "Modeling BPEL4WS," CodeGear Conference Proceedings, Borland, 2004.
- [119] "Unified Modeling Language," Object Management Group, <http://www.uml.org/>.
- [120] D. Braun, J. Sivils, A. Shapiro, and J. Versteegh, "Unified Modeling Language (UML) Tutorial," Kennesaw State University, 2001.
- [121] M. Fowler and K. Scott, *UML Distilled*. Addison-Wesley, 2000.
- [122] M. Chitnis, P. Tiwari, and L. Ananthamurthy, "Activity Diagram in UML."
- [123] N. Russell, W. M. P. v. d. Aalst, A. H. M. t. Hofstede, and P. Wohed, "On the suitability of UML 2.0 activity diagrams for business process modelling," in *3rd Asia-Pacific conference on Conceptual modelling*, Hobart, Australia, 2006.
- [124] S. Agarwal, S. Handschuh, and S. Staab, "Surfing the Service Web," in *Second International Semantic Web Conference (ISWC2003)*, Sanibel Island, Florida, 2003.
- [125] A. Ankolekar, M. Burstein, J. R. Hobbs, O. Lassila, D. L. Martin, S. A. McIlraith, S. Narayanan, M. Paolucci, and T. Payne, "DAML-S: Semantic Markup for Web Services," The DAML Services Coalition.
- [126] M. Paolucci, N. Srinivasan, K. P. Sycara, and T. Nishimura, "Towards a Semantic Choreography of Web Services," in *International Conference on Web Services*, Las Vegas, Nevada, USA, 2003.
- [127] D. J. Mandell and S. A. McIlraith, "Adapting BPEL4WS for the Semantic Web: The Bottom-Up Approach to Web Service Interoperation," in *Second International Semantic Web Conference (ISWC2003)*, Sanibel Island, Florida, 2003.
- [128] "IBM," www.ibm.com.
- [129] "Microsoft Corporation," www.microsoft.com.
- [130] "BEA Systems, Inc.," <http://www.oracle.com/bea/index.html>.
- [131] "SAP - Business Software Solutions Applications and Services," www.sap.com.
- [132] "Siebel Systems, Inc.," <http://www.oracle.com/applications/crm/siebel/index.html>.
- [133] E. Oren, "WSMO Deliverable - D13.2 v0.1 - WSMX Execution Semantics," DERI 2004.

-
- [134] S. Thatte, "XLANG, Web Services for Business Process Design," http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm; Microsoft Corporation, 2001.
 - [135] D. F. Leymann, "Web Services Flow Language (WSFL 1.0)," IBM Software Group, 2001.
 - [136] "Samples for ActiveBPEL 3.x," Active Endpoints, 2007.
 - [137] P. Wohed, W. M. P. v. d. Aalst, M. Dumas, and A. H. M. t. Hofstede, "Pattern Based Analysis of BPEL4WS," Queensland University of Technology, Brisbane 2002.
 - [138] M. A. Aslam, S. Auer, and M. Böttcher, "From BPEL4WS Process Model to Full OWL-S Ontology," in *Posters and Demos 3rd European Semantic Web Conference (ESWC 2006)*, Budva, Montenegro, 2006.
 - [139] H. Guo and X. Lin, "A Study on the an Application Integration Integrated Model Supporting Inter-enterprise Collaborations," in *Computer Supported Cooperative Work in Design II: Springer Berlin / Heidelberg*, 2006.
 - [140] Lee, Yoon, and Shin, "Supporting Dynamic Workflows in a Ubiquitous Environment " in *International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*, 2007.
 - [141] I. Horrocks, F. v. Harmelen, P. Patel-Schneider, T. Berners-Lee, D. Brickley, D. Connolly, M. Dean, S. Decker, D. Fensel, R. Fikes, P. Hayes, J. Heflin, J. Hendler, O. Lassila, D. McGuinness, and L. A. Stein, "DAML+OIL," 2001.
 - [142] K. K. Breitman, M. A. Casanova, and W. Truszkowski, "Semantic Web: Concepts, Technologies and Applications," Springer London, 2007.
 - [143] J. Peer and M. Vukovic, "A Proposal for a Semantic Web Service Description Format," in *European Conference on Web Services*, 2004.
 - [144] M. Sabou, D. Richards, and S. V. Splunter, "An experience report on using DAMLS," in *Twelfth International World Wide Web Conference Workshop on E-Services and the Semantic Web*, 2003.
 - [145] R. Lara, A. Polleres, H. Lausen, D. Roman, J. d. Bruijn, and D. Fensel., "A Conceptual Comparison between WSMO and OWL-S. WSMO Deliverable D4.1v0.1, 2005. <http://www.wsmo.org/2004/d4/d4.1/v0.1/>," 2005.
 - [146] "The Web Service Modeling Language WSML," ESSI WSMO working group 2005.
 - [147] "Web Service Modelling Ontology (WSMO)," ESSI WSMO working group.
 - [148] M. Moran and A. Mocan, "WSMX – An Architecture for Semantic Web Service Discovery, Mediation and Invocation," in *3rd International Semantic Web Conference (ISWC2004)* Hiroshima, Japan, 2004.
 - [149] H. Lausen, J. d. Bruijn, A. Polleres, and D. Fensel, "WSML - a Language Framework for Semantic Web Services," Position Paper for the W3C rules workshop 2005.
 - [150] M. Gone and S. Schade, "Towards Semantic Composition of Geospatial Web Services – Using WSMO in Comparison to BPEL," in *GI-Days 2007* University of Münster, Germany, 2007.
 - [151] A. Mocan, M. Kerrigan, and M. Zaremba, "Applying Semantics to Service Oriented Architectures," in *Oasis Symposium 2006* San Francisco, 2006.
 - [152] D. Kuropka and M. Weske, "Towards a service composition and enactment platform," *Int. J. Business Process Integration and Management*, vol. 2, 2007.
 - [153] P. Traverso and M. Pistore, "Automated Composition of Semantic Web Services into Executable Processes," in *3rd Int. Semantic Web Conf.*, 2004.
 - [154] B. Bordbar, G. Howells, M. Evans, and T. Staikopoulos, "Model Transformation from OWL-S to BPEL via SiTra," in *European Conference on Model Driven Architecture Foundations and Applications*, 2007.
 - [155] M. A. Aslam, S. Auer, J. Shen, and M. Herrmann, "Expressing Business Process Models as OWL-S Ontologies," in *Workshop on Grid and Peer-to-Peer Based Workflows (GPWW 2006)*, 2006.
 - [156] "Eclipse - an open development platform."
 - [157] "Process Specification Language PSL," <http://www.mel.nist.gov/psl/>; National Institute of Standards and Technology, U.S. Department of Commerce, 1999.
 - [158] L. Sterling, E. Shapiro, and R. Garrett, "The Art of Prolog," *IEEE Expert*, 1987.
 - [159] L. v. Bertalanffy, *General System Theory*; George Braziller Publications, 1968.
 - [160] W. R. Ashby, *Introduction to Cybernetics*. London: Chapman & Hall, 1958.

-
- [161] F. Heylighen and C. Joslyn, "What is Systems Theory?," Principia Cybernetica Web (Principia Cybernetica, Brussels), 1992.
 - [162] S. LAKMAZAHARI, "Constraint-based reasoning via Grobner Bases," *Artificial intelligence for engineering design, analysis and manufacturing*, 1997.
 - [163] E. C. Freuder and A. K. Mackworth, *Constraint-Based Reasoning*. MIT / Elsevier, 1994.
 - [164] J. Francis, "BPM and Nonlinear Thinkers," *Managing BPM*, 2004.
 - [165] J. Crampton, "On the satisfiability of authorization constraints in workflow systems," 2004.
 - [166] K. Verma, R. Akkiraju, R. Goodwin, P. Doshi, and J. Lee, "On Accommodating Inter Service Dependencies in Web Process Flow Composition," in *2004 AAAI Spring Symposium*, 2004.
 - [167] G. A. Bolcer and R. N. Taylor, "Advanced Workflow Management Technologies," Information and Computer Science, University of California, Irvine, 2008.
 - [168] F. Maurer, B. Dellen, F. Bendeck, S. Goldmann, H. Holz, B. Kotting, and M. Schaaf, "Merging Project Planning and Web-Enabled Dynamic Workflow Technologies," *IEEE Internet Computing*, 2000.
 - [169] D. L. Dean, R. E. Orwig, and D. R. Vogel, "Facilitation Methods for use with EMS Tools to Enable Rapid Development of High Quality Business Process Models," in *29th Annual Hawaii International Conference on System Sciences*, 1996.
 - [170] K. Balasubramanian, A. Gokhale, G. Karsai, J. Sztipanovits, and S. Neema, "Developing Applications Using Model-Driven Design Environments," *IEEE Computer Society*, 2006.
 - [171] D. Fahland, "Complete Abstract Operational Semantics for the Web Service Business Process Execution Language," Humboldt-Universität zu Berlin, Institut für Informatik 2005.
 - [172] M. M. Lehman and J. F. Ramil, "Role and Impact of Feedback and System Dynamics in Software Evolution Processes and their Improvement," Copenhagen, Denmark 2000.
 - [173] B. Karakostas, Y. Zorgios, and C. C. Alevizos, "Automatic derivation of BPEL4WS from IDEF0 process models," *Journal of Software & System Modeling, Springer Berlin / Heidelberg*, vol. 5, pp. 208-218, 2006.
 - [174] M. Owen and J. Raj, "BPMN and Business Process Management," Popkin Software, 2004.
 - [175] "Applications of Model-based Management," Microsoft Corporation 2005.
 - [176] "Standard Evaluations," Workflow Patterns Initiative, <http://www.workflowpatterns.com>, 2007.
 - [177] B. Karakostas, Y. Zorgios, and C. C. Alevizos, "The Semantics of Business Service Orchestration," in *Advances in Semantics for Web services 2006 Workshop (semantics4ws'06)*, Vienna, Austria, 2006.
 - [178] "Process Definition Interface- XML Process Definition Language, Document Number WPMC-TC-1025, Version 1.0,," Workflow Management Coalition, 2005.
 - [179] G. Shegalov, M. Gillmann, and G. Weikum, "XML-enabled workflow management for e-services across heterogeneous platforms," *The VLDB Journal — The International Journal on Very Large Data Bases*, 2001.
 - [180] W3C-XML, "Extensible Markup Language (XML)," <http://www.w3.org/XML>.
 - [181] W3C-XMLSchema, "XML Schema," <http://www.w3.org/2001/XMLSchema>.
 - [182] "Athens University of Economics and Business," <http://www.aueb.gr/>.
 - [183] "The ActiveBPEL® engine," <http://www.active-endpoints.com/active-bpel-engine-overview.htm>: Active Endpoints.
 - [184] "bexee - BPEL Execution Engine," Berne University of Applied Sciences, School of Engineering and Information Technology, <http://bexee.sourceforge.net/>: Active Endpoints.
 - [185] "Intalio|BPMS," <http://bpms.intalio.com/>: Intalio.
 - [186] "Java Technology," <http://java.sun.com/>, Sun Microsystems, Inc.
 - [187] A. Azim, A. Ghani, K. T. Wei, G. M. Muketha, and W. P. Wen, "Complexity Metrics for Measuring the Understandability and Maintainability of Business Process Models using Goal-Question-Metric (GQM)," *IJCSNS International Journal of Computer Science and Network Security*, vol. 8, pp. 219-225, 2008.
 - [188] J. Cardoso, "Process control-flow complexity metric: An empirical validation," in *IEEE International Conference on Services Computing*, Washington, DC, USA, 2006.

-
- [189] J. Cardoso, J. Mendling, G. Neumann, and H. A. Reijers, "A Discourse on Complexity of Process Models," in *Business Process Management Workshops*, 2006.
 - [190] J. Cardoso, "About the Data-Flow Complexity of Web Processes," in *6th International Workshop on Business Process Modeling, Development, and Support: Business Processes and Support Systems: Design for Flexibility*, Porto, Portugal, 2005.
 - [191] I. Vanderfeesten, H. A. Reijers, J. Mendling, W. M. Aalst, and J. Cardoso, "On a Quest for Good Process Models: The Cross-Connectivity Metric," in *20th International Conference on Advanced Information Systems Engineering*, Berlin, Heidelberg, 2008.
 - [192] H. A. Reijers and I. T. P. Vanderfeesten, "Cohesion and coupling metrics for workflow process design," in *2nd International Conference on Business Process Management (BPM 2004)*, 2004.
 - [193] J. Cardoso, "Evaluating Workflows and Web Process Complexity," in *Workflow Handbook 2005*, L. Fischer, Ed. FL, USA: Future Strategies Inc.: Lighthouse Point, 2005, pp. 284-290.
 - [194] J. Cardoso, "Complexity Analysis of BPEL Web Processes," *Software Process: Improvement and practice*, 2006.
 - [195] K. B. Lassen and W. M. P. v. d. Aalst, "Complexity metrics for Workflow nets," *Information and Software*, vol. 51, pp. 610-626, 2008.
 - [196] V. Gruhn and R. Laue, "Approaches for Business Process Model Complexity Metrics," *Technologies for Business Information Systems*, pp. 13-24, 2007.
 - [197] J. Mendling, "Testing Density as a Complexity Metric for EPCs," Vienna University of Economics and Business Administration, Vienna, Austria 2006.
 - [198] J. Cardoso, "Business Process Quality Metrics: Log-Based Complexity of Workflow Patterns," in *OTM Conferences*, Vilamoura, Portugal, 2007.
 - [199] K. Swenson, "Human Process: Trouble Ticket," <http://kswenson.wordpress.com/2008/01/01/human-process-trouble-ticket/>, 2008.
 - [200] K. Pijanowski, "What is Windows Workflow Foundation?," <http://www.keithpij.com/Home/tabid/36/EntryID/18/Default.aspx>, 2008.
 - [201] A. Belychok, "Humans Swimming In The Intalio Pool," <http://mainthing.ru/tag/bpmn/>, 2009.
 - [202] J.-J. Dubray, "Where is BPEL 2.0 going?," http://www.ebpml.org/bpel_2_0.htm, 2004.
 - [203] "POMPEI: P2P location and presence mobile services for crisis management," Programme: Sixth Framework Programme (FP6); IST Priority, Europe's Information Society, 2006-2007.
 - [204] H. He, "What is Service-Oriented Architecture?," <http://webservices.xml.com/pub/a/ws/2003/09/30/soa.html>, 2003.

Appendix – Performance of inference engine

We undertook several real and simulated tests to test the performance of the workflow inference engine, introduced in Section 4.5 of Chapter 4. There are four distinct phases:

- **Phase 1**: Based on a user's action or a request from a software agent, the engine checks the state of the requested Activity.
E.g. Execute (ActCodename: CalculateTaxAmount,
oInput: Invoice,
oProcess: Taxes)
At this phase, the engine checks whether the state of Activity *CalculateTaxAmount* is 'available'.
- **Phase 2**: Engine executes the requested Activity, based on its defined execution Mechanisms.
- **Phase 3**: After execution, the status of Activity is set to 'executed'.
- **Phase 4**: The engine checks all Activities of the process and for those whose status is 'available' the program flow returns to Phase 1 and so on.

Obviously, the most time consuming phases are 2 and 4. We ran our experiments on a COMPAQ PROLIANT server with dual XEON processors at 2.8GHz, 1-GB of RAM, and under a MS [129] Windows Advanced Server operating system. We varied different parameters including the number of Activities involved in a process, the number of Controls, Inputs, and execution Mechanisms of Activities. The results of performance are as follows.

Phase 1

Phase 1 is not CPU-intensive and is independent of the number of total Activities involved in the whole process model. It is also independent of the number of Controls, Inputs, and execution Mechanisms. The execution time was less than 3 ms (Figure 0.1).

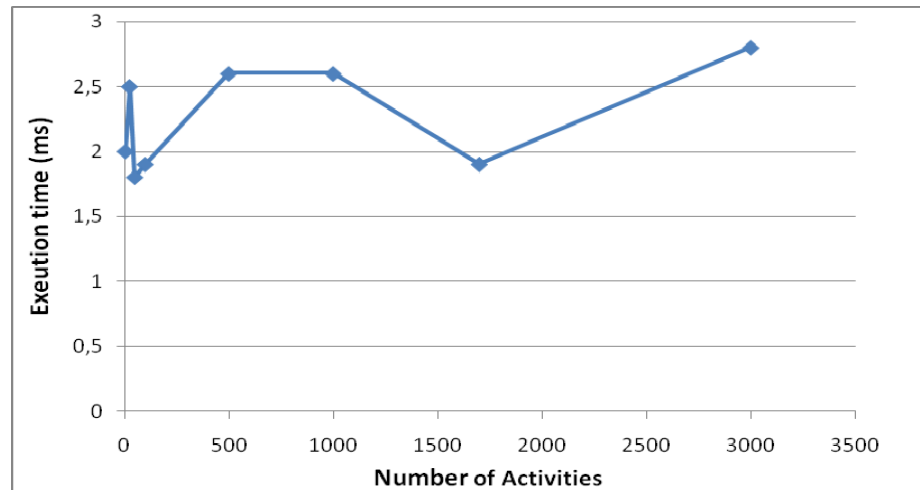


Figure 0.1: Phase 1's execution time (ms)

Phase 2

Phase 2 is also independent of the number of total Activities involved in the whole process model. It is dependent though on the number of execution Mechanisms of the Activity and relies on the execution performance of components and XSLT transformations. A typical Activity has 1 component type of mechanism or 1 XSLT type of mechanism. In extreme cases, an Activity may have more mechanisms. In Figure 0.2 the results show that execution time for a typical Activity varies from 11ms to 19ms.

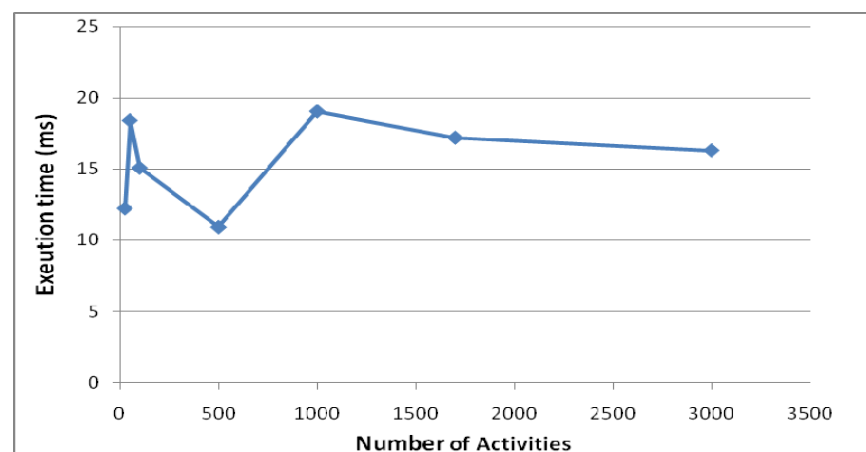


Figure 0.2: Phase 2's execution time (ms) for a typical Activity

On the other hand, in extreme cases where an Activity has more than 2 execution Mechanisms we observe that the execution time is exponential as shown in Figure 0.3.

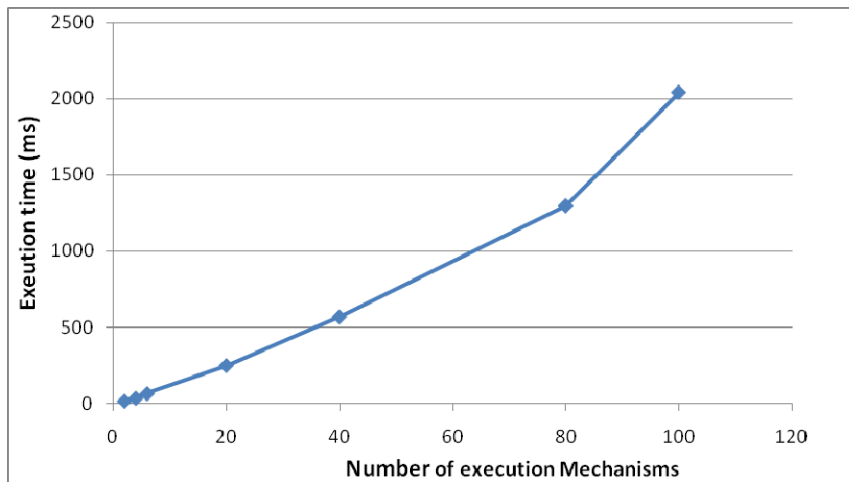


Figure 0.3: Phase 2's execution time (ms) depending on execution Mechanisms

Phase 3

Phase 3 is not CPU-intensive and is independent of the number of total Activities involved in the whole process model. It is also independent of the number of Controls, Inputs, and execution Mechanisms. The average time was less than 1 ms (Figure 0.4).

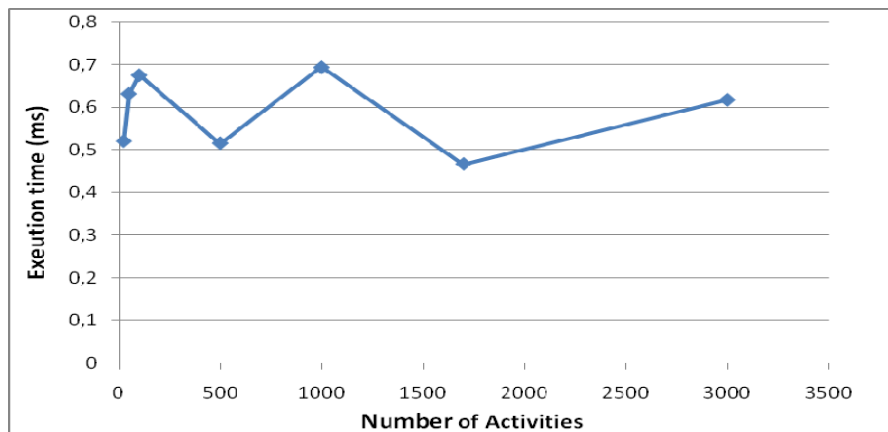


Figure 0.4: Phase 3's execution time (ms)

Phase 4

Phase 4 may require analysis of large and complex process models in order to check which Activities of the process are 'available'. The performance of this phase clearly depends on the number of Activities of the process. The results show that even in the extreme scenario of having 3000 Activities in a single process model the average time is acceptable, at around 2 seconds (2232 ms) as depicted in Figure 0.5.

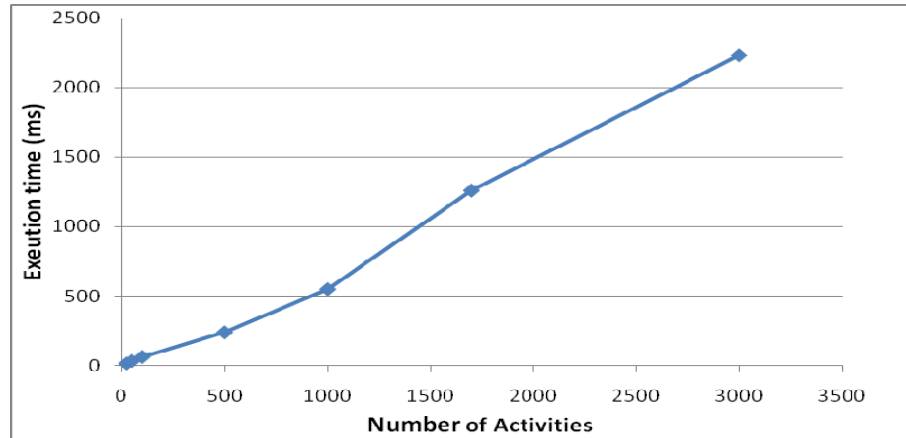


Figure 0.5: Phase 4's execution time (ms)

Apart of the number of Activities, performance of Phase 4 is slightly dependant on the number of Inputs and Controls of the Activities. Figure 0.6 shows the performance of checking an Activity's state, based on the number of Inputs and Figure 0.7 shows the performance of checking an Activity's state, based on the number of Controls.

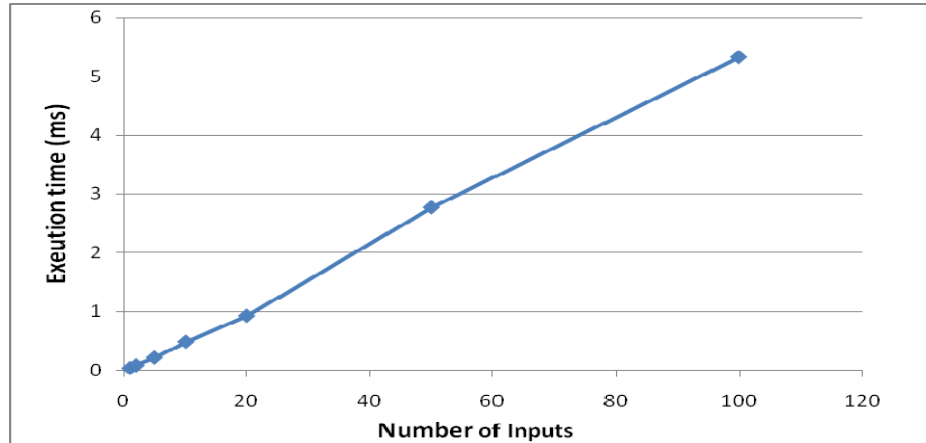


Figure 0.6: Phase 4 – Execution time (ms) based on the number of Inputs

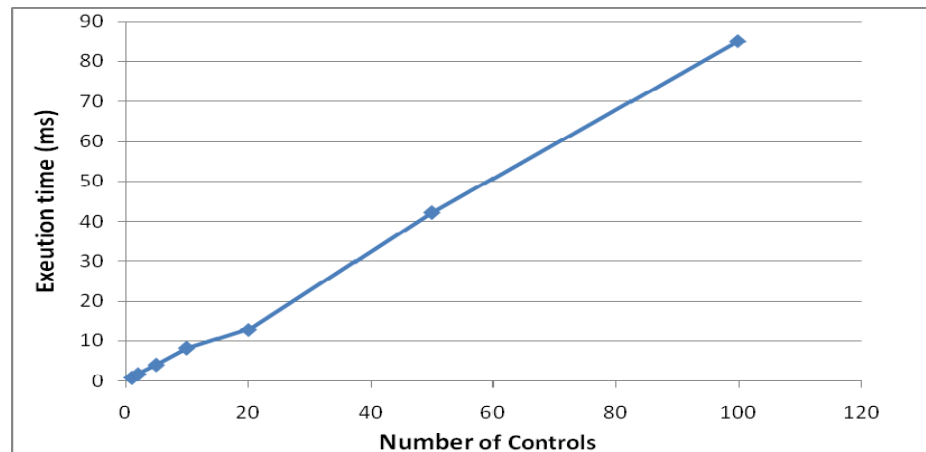


Figure 0.7: Phase 4 – Execution time (ms) of checking an Activity's state

We observe therefore that the number of Controls in Activities affect the performance more than the number of Inputs. Having 100 Inputs in an Activity, which we consider as

an extreme case, results in a performance of about 5ms (Figure 0.6), while having 100 Controls, which we also consider an extreme scenario, results in a performance of about 85ms (Figure 0.7). This is logically, as in case of Controls the engine needs to evaluate each one of them, while in case of Inputs the engine just checks if they are available or not.

Overall, the total execution time is represented in Figure 0.8, using a common scenario where Activities have an average number of 2 execution Mechanisms, 4 Inputs and 3 Controls. In this case, we consider the average performance of 645,95ms reasonable and acceptable.

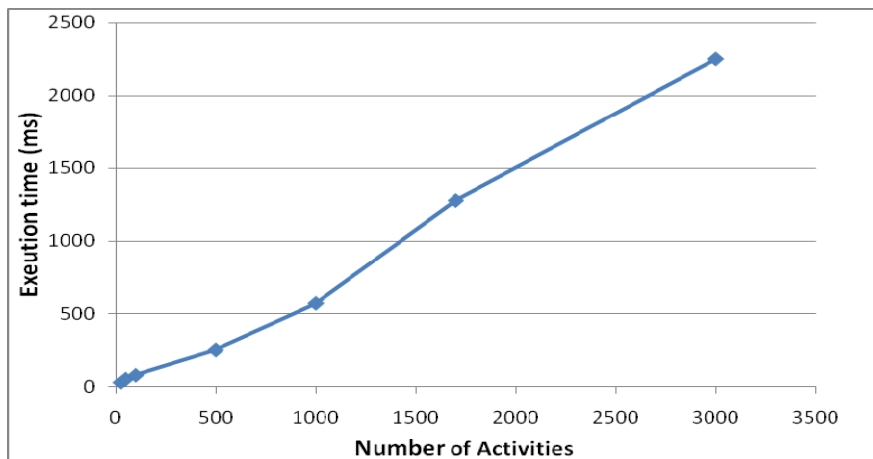


Figure 0.8: Total Execution time (ms) using a typical scenario

Figure 0.9 also shows the total execution time of an extreme case where Activities have an average number of 10 execution Mechanisms, 40 Inputs and 3 Controls. In this extreme case the average performance of 13359ms is expected but in the same time not acceptable for a real working system.

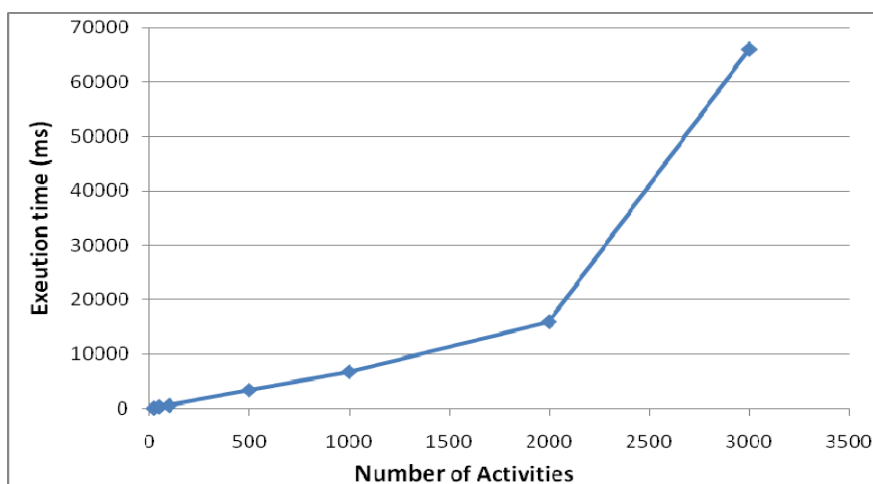


Figure 0.9: Total Execution time (ms) using an extreme scenario