



City Research Online

City St George's, University of London

Citation: Netkachova, K., Netkachov, O. & Bloomfield, R. E. (2015). Tool Support for Assurance Case Building Blocks, Providing a Helping Hand with CAE. Paper presented at the SAFECOMP 2015 Workshops, ASSURE, DECSoS, ISSE, ReSA4CI, and SASSUR, 22-09-2015, Delft, Netherlands. doi: 10.1007/978-3-319-24249-1_6

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/12968/>

Link to published version: https://doi.org/10.1007/978-3-319-24249-1_6

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](#).

Tool Support for Assurance Case Building Blocks

Providing a Helping Hand with CAE

Kateryna Netkachova¹², Oleksandr Netkachov¹² and Robin Bloomfield¹²

¹Centre for Software Reliability, City University London, UK
{Kateryna.Netkachova.2, Oleksandr.Netkachov,
R.E.Bloomfield}@city.ac.uk

²Adelard LLP, London, UK
{kn, reb}@adelard.com

Abstract. This paper presents a tool for structuring arguments in assurance cases. The tool is designed to support the methodology of Claims-Arguments-Evidence (CAE) Building Blocks that provides a series of archetypal CAE fragments to help structure cases more formally and systematically. It assists with the development and maintenance of structured assurance cases by providing facilities to manage CAE blocks and partially automate the generation of claim structures. In addition to the tool, new visual guidelines called “Helping hand” is provided to assist in applying the building blocks. The tool has been implemented on the Adelard ASCE platform. The target users are assurance case developers and reviewers. The tool and associated methodology can also be useful for people learning how to structure cases in a more rigorous and systematic manner.

Keywords: Claims·argument·evidence·CAE building blocks·helping hand·ASCE tool·support.

1 Introduction

Over the past ten years there has been a trend towards an explicit claim-based approach to safety justification and considerable work has been done on developing and structuring assurance cases [1,2,3]. However, the practice of how to structure and present cases is very varied. There are lots of different styles with different expressiveness and these many approaches make it difficult to compare cases and hard to provide a more rigorous semantics. To address these issues and provide a more rigorous approach to architecting cases, we have defined specific rules that restrict the type of argument structures and developed a collection of building blocks for assurance cases that help construct cases more formally and systematically [4].

During the development of CAE building blocks, we reviewed a wide range of cases from the defence, medical, financial and nuclear sector and the proposed set of building blocks were able to capture most of what was being expressed. We wish to deploy these CAE building blocks, evaluating their use and improving the methodology.

The tool presented in this paper is designed to aid the research and practice of developing structured formal and semi-formal assurance cases. There are other products [5,6] available to assist in the structured assurance case development. What makes our tool unique is support for the CAE blocks as self-contained reusable configurable components. It is a purpose-built tool designed specifically for the building blocks methodology, therefore, it was essential to integrate it with a widely-used assurance case software to make an impact. We implemented it on top of ASCE [7], which is a market-leading tool for the development and maintenance of assurance cases across a wide range of industries. ASCE is a commercial product but it is available free of charge for academic research purposes.

The paper is structured in the following way. The concept of CAE building blocks needed to understand the idea and a new “helping hand” guidance are introduced in Section 2. The software tool, which is the main focus of the paper, along with its technical information and implementation details are described in Section 3. Some early experience with the tool and the future directions of work are outlined in Section 4.

2 CAE Building Blocks and the “Helping Hand”

2.1 Building Blocks Concept

CAE building blocks are a series of archetypal CAE fragments, derived from an empirical analysis of real cases in various domains. They are created using a standardised structure for combining CAE and are part of a stack of resources that we are developing to support authors of assurance cases. These resources comprise the basic concepts of claims, argument, evidence; building blocks with a set of specific CAE structures; templates created out of the blocks to address particular classes of problems, and the overall assurance case created using blocks and templates. The stack of CAE resources is shown in Fig. 1, where arrows indicate the instantiation of elements to produce an assurance case. The approach can be extended to support GSN notation [3] as well. In that case, GSN elements will be used instead of CAE and GSN patterns will be constructed out of the building blocks in a similar way as the CAE templates. This extension will be implemented in due course.

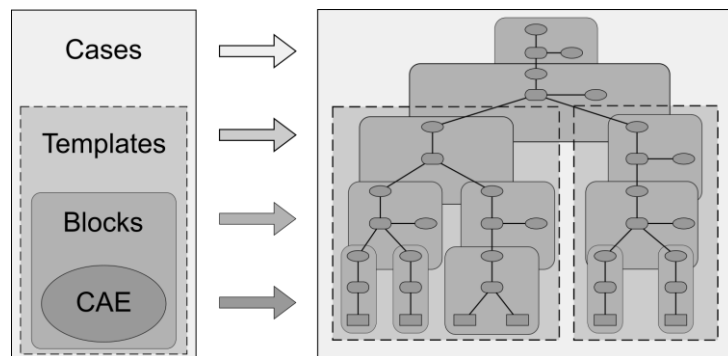


Fig. 1. Schematic of the stack of CAE resources (left) instantiated into a specific case (right)

The block structure contains enhancements to the classical CAE approach [1,2]. One enhancement is to how arguments are addressed: a special side-warrant element is introduced to explain and assist in a structured way whether the top-level claim can be deduced from the subclaims and under which circumstances the argument is valid. The five basic CAE building blocks that we have identified are:

- Decomposition – partitions some aspect of the claim
- Substitution – refines a claim about an object into another claim about an equivalent object
- Concretion – gives a more precise definition to some aspect of the claim
- Calculation or proof – used when some value of the claim can be computed or proved
- Evidence incorporation – incorporates evidence that directly supports the claim

The summary and the structure of these basic block are provided in the Appendix A. Additional information and guidance can be found in the paper [4].

2.2 “Helping Hand” for CAE Building Blocks

In order to support the teaching and deployment of CAE Building Blocks, we have created a visual guidance shown in Fig. 2. We call it a “helping hand” as it is designed to help people structure assurance cases in an easier and more intuitive way by providing a “cheat sheet” on a hand with some hints and questions to answer. Instead of wondering what to do next and how to better expand the case, this approach shifts the question to an easier one: “which block is best to use?” and helps to find the answer by following the provided guidance.

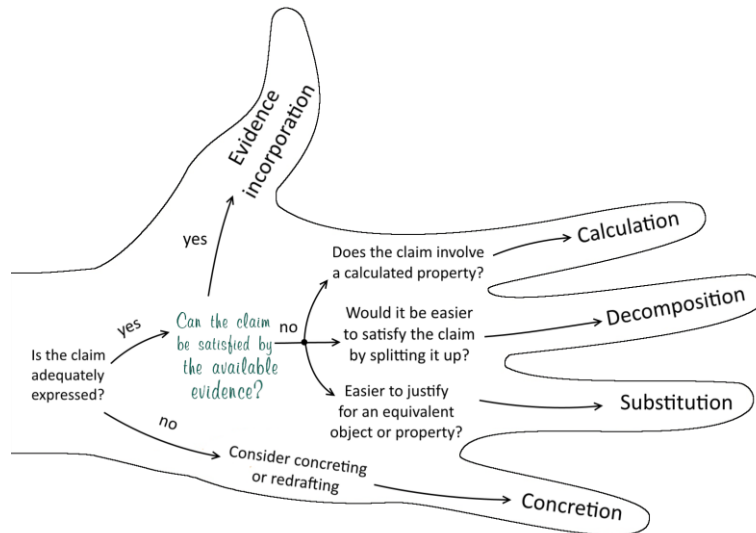


Fig. 2. “Helping hand” – high level guidelines for applying the building blocks

3 Tool Description

The main focus of this paper is on the software tool that assists in using CAE building blocks within the existing assurance case development processes. The tool we have developed provides facilities for creating and managing block-based argumentation to help create more formal, structured and maintainable assurance cases.

The usage of CAE Building Blocks is not isolated and in order to be effective our tool should be integrated with the current processes and other tools used for the creation and management of cases. To address this, we implemented it on top of the ASCE platform, which is a widely-used powerful graphical and narrative hypertext tool for the development, review and maintenance of assurance cases. The detailed description of the ASCE tool can be found in the help file [8]. Below we only highlight the features of ASCE that are used by our tool and needed to understand the rest of the paper.

- Graphical editor for creating and arranging arguments
- Support for different notations, including CAE
- A content editor for editing the narrative content of nodes in a HTML format
- Functionality to validate the resulting network against the logical constraints of the notation being used
- Extensibility allowing support for specific applications and integration with other technologies

The extensibility feature is particularly important for us as it is used to incorporate our tool into the existing ASCE environment. The integration is performed through the use of the ASCE mechanism of plugins and a customised “schema” file.

Therefore, the implementation of the tool involved two major activities: supporting the Building Blocks methodology and integrating it into the ASCE tool. Each of them is described in sections 3.1 and 3.2 below. The interaction between ASCE and both parts of our tool is schematically shown in the sequence diagram provided in Fig. 3.

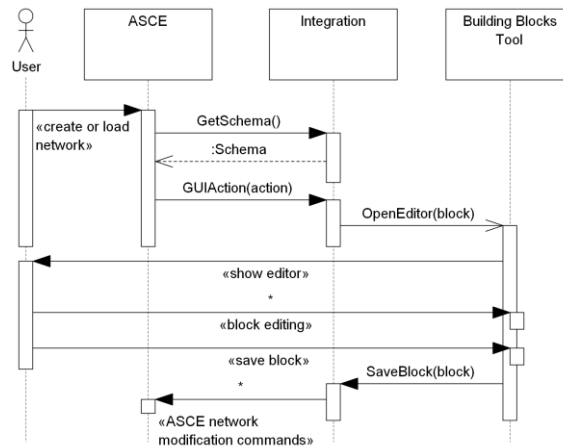


Fig. 3. Interaction sequence diagram

3.1 Tool Support for CAE Building Blocks

The Building Blocks tool is developed as a DHTML application. The graphical user interface (GUI) components are created using HTML5, CSS3 and JavaScript. The structure of the GUI controls follows the Model-View-Controller architectural pattern: every control has a model containing its internal state, HTML view reacting to any changes of that model and controllers reacting to the user events and modifying the model based on them. Examples of the GUI for the decomposition and concretion blocks are provided in Fig. 4.

Most of the fields are completed automatically to save the user unnecessary typing. For example, the top claim is parsed to locate the name of the object. As soon as the values are completed, the subclaims titles, argument and side-warrant text are generated. One of the design choices we made is the ability to grade the formality, e.g. the side-warrant can be formulated informally or it can be generated by the tool in a more formal way (math based side-warrant). All the inputs are editable and the users are free to alter the text of the subclaims, side-warrants, etc. the way they want. The OK button at the bottom of the dialog is used to apply the block. Of course, it is not a one way write as the tool supports the evolution of CAE structures. If the users decide to change their minds and delete or modify the nodes, this is reflected back in the tool. In that case the automatically created text is regenerated, while any custom modifications are preserved (no user data is lost). A sample CAE structure and the tool GUI with dependencies between auto-generated text values shown in red are provided in Fig. 4

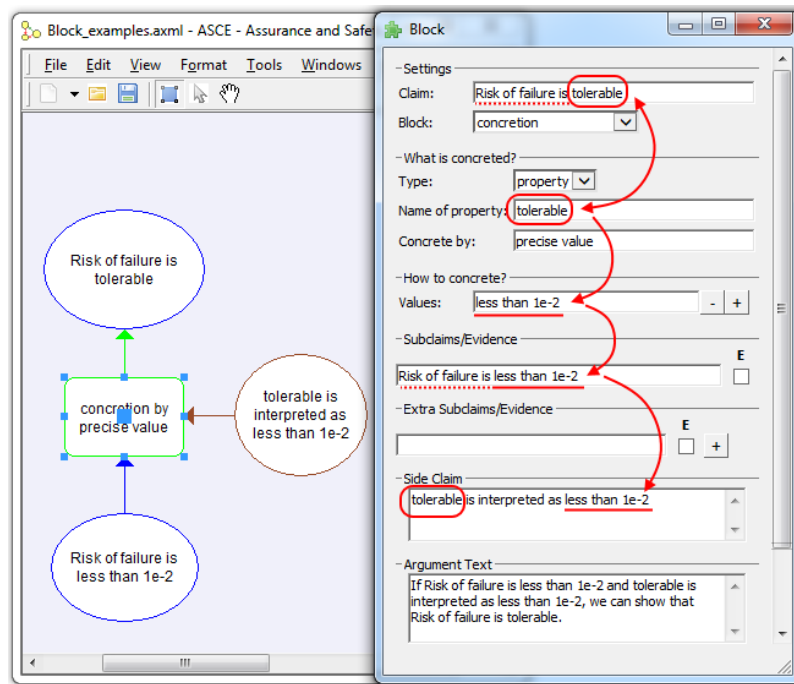


Fig. 4. GUI and sample CAE diagram for the concretion block

In terms of the implementation details, the following JavaScript libraries are used by the GUI components: jQuery for DOM querying and manipulating, Backbone for implementing the Observer pattern, Lo-Dash for general-purpose object model queries. The standard HTML controls such as inputs and checkboxes are wrapped by the MVC triads to keep GUI control set consistent. In addition to those wrappers, there are custom application-specific controls, such as ListControl, which iterates over the collection of items, rendering each of them into an independent row. The class diagram for the tool is provided in Fig. 5.

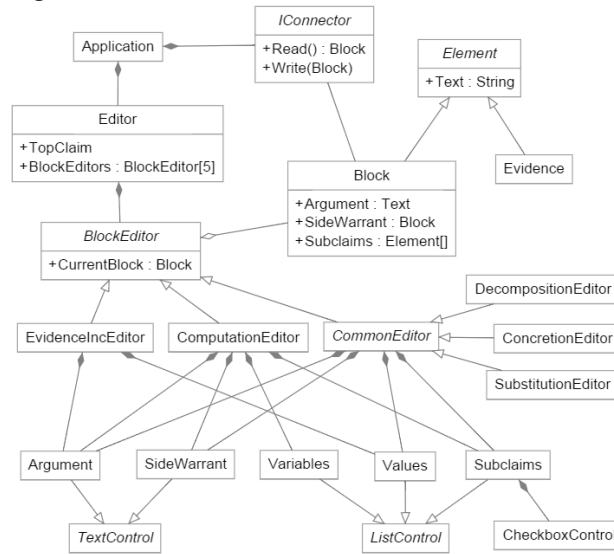


Fig. 5. UML class diagram for the tool

All classes of the tool are grouped into three main packages:

- Block model - classes that represent Block elements
- GUI - user-interaction controls and Block editors, constructed from these controls
- Application engine - manages instantiation of Block editors, contains dependency injection points for connector used for the integration with ASCE

The classes representing CAE Building Block elements (claims, argument, side-warrant etc.) and links are included in the Block model package. This package also contains rules for checking whether the model is well-formed by following the rules of the CAE normal form, specifically:

- Claim nodes may only be connected to argument nodes
- Argument nodes may only be connected to claim and evidence nodes
- Each argument node may only have one outbound link to a claim node
- Each claim is to be supported by only one argument
- Argument nodes must be supported by at least one subclaim or evidence node
- Evidence nodes represent the bottom of the safety argument and are not supported
- A claim, subclaim or evidence may support more than one argument

All modules within the packages conform to the CommonJS Modules specification. To load these modules the execution environment should contain the implementation of the CommonJS “require” function. The next section describes the integration part, where this function is implemented by using the Windows Scripting Host components.

3.2 Integration with ASCE

The extensible architecture of ASCE allows users to implement new features on top of the core functionality of ASCE using additional schemas and plugins. The developer documentation is freely available and can be found at [9,10]. Basically, ASCE plugins are written as XML files which contain a mixture of configuration information, user interface and code. The recommended approach is to use HTML forms with event handlers created in one of the Windows Scripting compatible languages (VBScript or JScript). The GUI approach used for our tool is suitable for this type of integration, so we implemented the tool as an ASCE plugin that runs in the Web browser component. The two major integration tasks we had to solve involved:

1. *Implementation of CommonJS API in the plugin using Windows Scripting Components:* As was mentioned above, ASCE uses Windows Scripting while our tool is built using CommonJS architecture. In order to use CommonJS with ASCE, we had to implement CommonJS interfaces using objects available in Windows Scripting.
2. *Implementation of the converter between the object models of the building blocks and ASCE:* Specific classes of the ASCE tool that are used by the converter are shown in the Fig. 6.

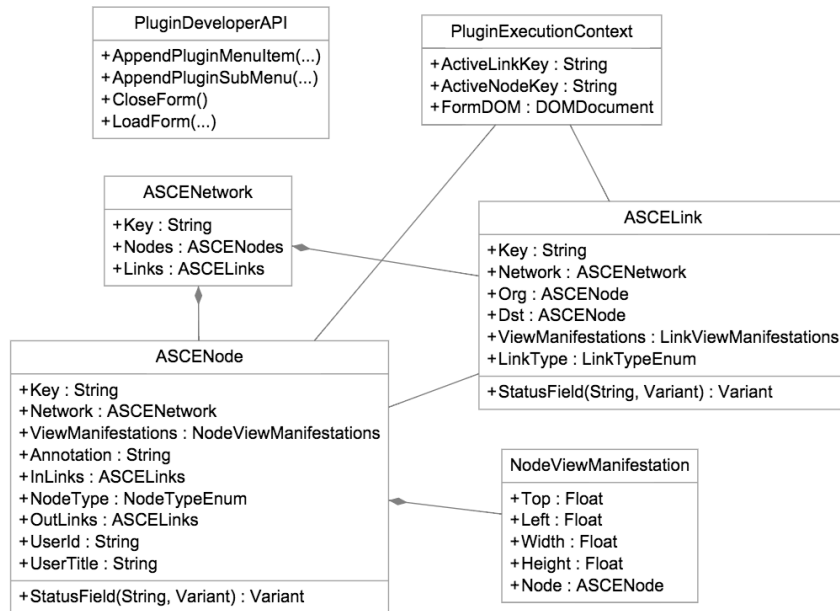


Fig. 6. Class diagram showing ASCE COM components

Additionally, we also created a new ASCE schema file with a few custom node properties used to store the block settings.

4 Conclusions and Future Directions

In this paper we have presented a software tool for structuring assurance cases using CAE Building Blocks. The tool is integrated in the ASCE environment through the use of additional schemas and plugins. Additionally, we have introduced a high level guidance – a “helping hand” – to assist in the case structuring process. The tool and the methodology are going through a progressive, iterative approach to deployment and will continue to evolve. At the moment, CAE notation forms the basic blocks of the approach. However, it can be extended to other graphical and tabular notations and their tool support in the future.

We have already deployed the prototype tool and the methodology on a number of projects. Some of the completed tasks include drafting of guidance for the IAEA on the assessment of dependability of nuclear I&C systems important for safety, drafting of templates for arguing about statistical testing as part of the EU Harmonics project, developing cases to address probabilistic modelling of critical infrastructure and particular how one addresses model doubt. We have also used CAE Blocks on a professional Masters level course at City University London on Information Security and Risk in an Assurance Case module.

The experience to date has shown the utility of the building blocks. However, there is more research and development to be done. For example, we need to explore composition of blocks into reusable domain-specific fragments or patterns, using GSN notation elements [3] and a related formal basis [11]. We also plan on looking into links to challenge and review checklists generated from the blocks, enhancing the default evidence incorporation block to be a composite block for trusted evidence and providing more support for the formal aspects of assurance cases. This is a very active and growing area with a number of research trends on argumentation, confidence and model based approaches and we plan to continue our research in this direction. In addition we will reflect on how the experience of CAE Blocks can further support Assurance Case workflows as well as what impact they might have on standardisation activities.

Acknowledgement

We acknowledge support from the Artemis JU SESAMO project (number 295354) and the UK EPSRC funded Communicating and Evaluating Cyber Risk and Dependencies (CEDRICS) project which is part of the UK Research Institute in Trustworthy Industrial Control Systems (RITICS).

A Appendix - Basic Building Blocks for Assurance Cases

Structure	Description
	Decomposition block This block is used to claim that a conclusion about the whole object or property can be deduced from the claims or facts about constituent parts.
	Substitution block This block is used when a claim needs to be given a more precise definition or interpretation. The top claim $P(X, Cn, En)$ can be replaced with a more precise or defined claim $P1(X1, Cn, En)$, Cn and En are configuration and environment.
	Concretion block This block is used when a claim needs to be given a more precise definition or interpretation. The top claim $P(X, Cn, En)$ can be replaced with a more precise or defined claim $P1(X1, Cn, En)$, Cn and En are configuration and environment.
	Calculation block This block is used to claim that the value of a property of a system can be computed from the values of related properties of other objects. Show that the value b of property $Q(X, b, E, C)$ of system X in env E and conf C can

	be calculated from values $Q_1(X_1, a_1, E, C), Q_2(X_2, a_2, E, C), \dots, Q_n(X_n, a_n, E, C)$
	Evidence incorporation block This block is used to incorporate evidence elements into the case. A typical application of this block is at the edge of a case tree where a claim is shown to be directly satisfied by its supporting evidence.

References

1. ISO/IEC 15026-2: Systems and software engineering -- Systems and software assurance -- Part 2: Assurance case, 2011.
2. Bishop, P.G, Bloomfield, R.E.: A Methodology for Safety Case Development. In: Safety-critical Systems Symposium 98, Birmingham, UK, Feb 1998, ISBN 3-540-76189-6.
3. Kelly, T.: The goal structuring notation-a safety argument notation. In: Proc. DSN 2004 Workshop on Assurance Cases, 2004.
4. Bloomfield, R.E., Netkachova, K.: Building Blocks for Assurance Cases. In: IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW) 2014, pp. 186-191, doi:10.1109/ISSREW.2014.72.
5. Denney, E., Pai, G., Pohl, J.: AdvoCATE: An Assurance Case Automation Toolset. In: F. Ortmeier, P. Daniel (eds.) SAFECOMP 2012 Workshops. LNCS, vol. 7613, pp. 8-21, Springer, Heidelberg, 2012.
6. Aiello, M., Hocking, A., Knight, J., Rowanhill, J.: SCT: A Safety Case Toolkit. In: IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW) 2014, pp. 216-219, doi:10.1109/ISSREW.2014.99.
7. Adelard LLP: Assurance and Safety Case Environment (ASCE). [Online]. Available: <http://www.adelard.com/asce/> [Accessed: 29 June 2015].
8. Adelard LLP, "Assurance and Safety Case Environment (ASCE) Help File". [Online]. Available: <http://www.adelard.com/asce/> [Accessed: 29 June 2015].
9. Emmet, L.: Introduction to plugin and schema development for ASCE. [Online]. Available: http://www.adelard.com/asce/plugins/developer-documentation/4.1/w1873v01c_ASCE_v4_plugin_API_docs.doc [Accessed: 29 June 2015].
10. Emmet, L.: API documentation for ASCE v4.1. [Online]. Available: http://www.adelard.com/asce/plugins/developer-documentation/4.1/w2082v02a_ASCE_plugin_developer_documentation.doc [Accessed: 29 June 2015].
11. Denney, E., Pai, G.: Formal Basis for Safety Case Patterns. In: 32nd International Conference on Computer Safety, Reliability and Security (SAFECOMP 2013), LNCS 8153, pp. 21-32. Sep. 2013.