SCT: A Safety Case Toolkit

M. Anthony Aiello, Ashlie B. Hocking, John Knight, Jonathan Rowanhill
Dependable Computing LLC
Charlottesville, VA USA

***Abstract*—SCT is a safety case toolkit designed to support the development and maintenance of safety cases for large, safety-critical systems. SCT supports safety case development by providing facilities to manage the file structure associated with the safety case, editors for various notations including GSN, and a build system that creates a custom web site to store the safety case. The web-based representation of the safety case includes a variety of features for safety case examination including comprehensive hyperlinking of elements, a GSN viewer, an argument index, and various custom reports.**

Keywords—Safety case, Goal Structuring Notation, safety case tools.

I. Introduction

In this paper, we present Dependable Computing's *Safety Case Toolkit* (SCT). The SCT is comprehensive system that supports the development, management, and presentation of system safety cases for large, safety-critical applications.

A safety case produced by SCT is a web site that is designed to operate as a stand-alone facility on a user's computer system (i.e., no network access is involved). SCT provides a set of services that allow developers to: (a) build and maintain the safety case and (b) manage the processes and synchronization necessary when teams work together. The toolkit provides facilities to produce the associated web site, and the web site presentation of a safety case provides the user with a number of unique features.

Most of the present SCT implementation is independent of the computing platform upon which the toolkit operates. The implementation operates on Apple Macintosh computers because of some dependencies on support packages. The safety case web sites can be accessed using any modern browser.

In Section II we discuss the requirements that SCT is designed to address. In Section III we summarize the features that SCT provides for developing safety cases. In Section IV we present the features in SCT that support the overall development process. In Section V we discuss the features provided in the safety cases that SCT generates. In Section VI we describe the primary user interface that SCT provides. In Section VII we present our conclusions.

II. Toolset Requirements

The requirements for SCT have evolved over approximately four years of prototype development and assessment. During this period of evolution, informed users have exposed prototype versions of SCT and safety cases built with SCT to a variety of demands. This process helped to reveal the requirements for SCT and refine the details as the toolkit was built.

This research was supported in part by NASA under NRA contract NNL13AA08C.

At the highest level, the requirements for the SCT are that the toolkit support:

- **Large teams**. A critical requirement for SCT was support for development programs involving large numbers of engineers, domain experts, and users. The definition of "large" in this case is large in every relevant dimension. Thus, the requirement includes support for large teams of safety case developers, large teams of subject-matter experts and technical-area experts, large teams of safety case users, and possibly even large teams of certifiers/regulators.
- **Large systems**. The types of system for which SCT is designed are large, complex, safety-critical systems involving significant numbers of technical domains.
- **Safety case development**. Users of a safety case require certain content, presentation formats, and services that influence creation extensively.
- **Extensive evidence**. Large systems require diverse, elaborate and voluminous evidence. Support for such bodies of evidence is clearly essential.
- **Management decision making**. A safety case can (and arguable should) act as a focal point for project management since material in the safety case provides a lot of material upon which management decisions are made.
- **Certification**. Any large safety-critical system will be subject to some form of audit, examination, or assessment prior to deployment.
- **Flexible argument manipulation**. Development of arguments for the safety cases of complex systems frequently requires extensive and protracted manipulation of arguments to facilitate the exploration of alternative lines of argumentation. Thus support is required for adding and updating argument fragments, and refactoring and reorganizing arguments.
- **High-quality visual representation**. Much of the material in a safety case relies on human insight and judgment for development and assessment. High-quality visual presentation helps to ensure accurate human interpretation.
- **Flexible presentation**. As well as high-quality visual presentation, human access to a safety case requires ease of navigation, comprehensive hyperlinking, and

the ability to adjust views so as to present material in an easily digestible form.

- **Process**. Many activities have to be undertaken by many different stakeholders in the development of a large safety case. The processes undertaken need to be synchronized, coordinated and managed with as much tool support as possible.

- **Asset access control**. A safety case might involve elements that cannot be universally accessible.

SCT has been developed to address these requirements. Some of the requirements are supported comprehensively, others less so, and access control not at all. Access control is assumed at present to be provided by the underlying operating system.

The toolkit has demonstrated its success in meeting these requirements in a number of safety case development projects including both large and small systems, and both production and experimental environments. At present, no empirical studies of the toolkit have been conducted to document performance in a statistically rigorous way.

III. Safety Case Development

The SCT supports development of safety cases using a set of services integrated into an Eclipse framework. The services allow management of all of the files used in a safety case together with access to various editors for the different file types.

Two important types of file are supported extensively:

- GSN arguments. GSN arguments can be manipulated in a variety of ways. A sophisticated GSN editor is provided.

- MultiMarkdown documents. Text files are used by the SCT to document descriptive material required for a subject safety case. MultiMarkdown (MMD) is a format designed to allow text to be created with a simple markup format so as to allow rapid text creation and subsequent translation into a presentation format. SCT translates MMD files to HTML for incorporation into the safety case web site.

A. Asset Management

The SCT asset manager is the interface that allows the user to organize the various assets in the safety case and invoke the associated editors.

The asset manager displays the assets in a typical Eclipse panel. Assets of a safety case include:

- GSN arguments.

- Text files that provide descriptive material for the body of the safety case.

- Figures that are needed to complement descriptive material.

- Attached files that contain material not processed by SCT but constitute part of the safety case, frequently items of evidence.

- Help files that contain material tailored to assist the user of the subject safety case.

B. GSN Editor

The SCT GSN editor allows rapid prototyping of argument ideas as well as supporting comprehensive development of argument details.

The primary features that support rapid prototyping of arguments are:

- Keyboard shortcuts for insertion of GSN nodes. A single key is used to add a node of a specific type to a selected node. Thus, for example, a simple argument consisting of a top-level goal, a context, a strategy, and several sub-goals requires a single menu-item selection to create a new argument with a default top-level goal, a letter “c” to create the context node, a letter “s” to create the strategy node, and one letter “g” for each sub-goal. All of the nodes will be created and laid out on the screen. Text can then be added to the nodes as desired using a simple inspector panel.

- Clicking and dragging to revise argument structures. Frequently, experimentation with different argument structures is desirable where sub-arguments need to be moved. The GSN editor supports movement of sub-arguments by merely selecting the top-level goal of the sub-argument and dragging it to the desired new location.

- Undo. Any editing operation can be undone if the change is deemed unnecessary.

The primary features that support comprehensive development of argument details are:

- Automatic argument layout. Arguments of arbitrary size are laid out automatically for the user immediately after a change is made. When arguments become large, context can easily be lost if the structure is not easily visible. Automatic argument layout ensures that the visual structure of the argument is always clear to the user.

- Comprehensive typed metadata. Frequently, arguments are dependent on or are associated with properties that are not inherently part of GSN. For example, GSN nodes might be classified according to their technical area, to the individual responsible for reviewing the node details, or the degree of confidence the user has in the node details.

  Such material can be represented easily as metadata in the GSN editor and displayed for the user or examined by a reporting system. Any number of typed metadata fields can be defined, and the type of the metadata set by the user as string, Boolean, integer, real, or enumerated.

  To facilitate the rapid association of a particular metadata value with a collection of nodes, a metadata value can be “stamped” onto a node quickly and conveniently with a single mouse click.

- Versioning support. In the development of a large safety case, maintaining versions of arguments and recording reasons for change are frequently required. The GSN editor provides comprehensive support for versioning either by requiring users to record the rationale for each change or by allowing batches of changes to be identified and the rationale for the batch to be recorded.

C. Build System

The SCT build system operates automatically based on the prescribed set of files and associated structures set up by the asset-management system to create the safety case web site.

All GSN nodes are given *unique* and *permanent* identifiers to allow each node to be located and referenced throughout the lifetime of the safety case during mechanical analysis. The build system updates the GSN unique identifiers to ensure that existing identifiers are properly maintained and required new identifiers are created properly.

Once the GSN identifiers are established, the build system renders all of the arguments into the required format including rendering of GSN to SVG, PDF and PNG, formats all of the text files into HTML, creates the argument index and all of the document items in the presentation format (see Section V.A).

The build system prepares a variety of standard and optionally custom reports for inclusion in the safety case web site. In particular, the build system formats and integrates a glossary of acronyms and a glossary of definitions that are tailored (manually) to the needs of the specific system for which the safety case is being created. The versioning data collected by the GSN editor can be output as a CSV file for subsequent custom processing. Finally, custom reports can be developed for the metadata used in arguments.

All of the arguments are integrated into the safety case web site by the build system. If desired, arguments can be placed as figures within text so as to allow descriptive material in the text to be located near the argument. Arguments in figures are supported by the review mechanism described in Section V.C.

Finally, the build system also executes a variety of tests on the safety case web site to check for avoidable defects that might arise.

IV. Development Process

A safety case is often developed as a shared resource amongst multiple teams across multiple domains and agencies. As a safety case is a living document, considerable lengths of time may pass between development cycles.

As a result, we apply several techniques to promote scalable and coordinated development of safety cases over time. Generally, these are in-practice technology from software development regulated by enforced workflow processes.

As explained in the previous section, the SCT has its own, complete build system. SCT uses Maven [2], a software build management system, to normalize this interface with existing externally available tooling.

For example, SCT uses Maven to define a safety case project. This approach allows easy integration of building, cleaning, and reporting about a safety case within the Eclipse editor. Any other tool capable of interacting with a Maven project can build and generate reports about a safety case.

SCT uses Git [3] for version control of a safety case project. Users can share safety cases through Git and build them using the Maven or native interface. Users can edit safety cases using the Eclipse editing package, and they can view the resulting safety case in a web browser. Combined, users can create an informal process of interaction to co-create and manage a safety case.

For larger projects, informal coordination and management can be infeasible or undesirable. To this end, the safety case toolkit includes the concept of a complete lifecycle of an assured system. A key component of the lifecycle is its ability to manage orderly processes for the development and maintenance of a safety case.

The SCT configures the lifecycle for a specific system to meet the needs of that particular safety assurance project. Projects are configured by choosing (and optionally tailoring) resources from a shared repository. These resources include libraries of GSN patterns, tools, guidance materials, and process frameworks.

Where guidance alone is insufficient, processes provide formally defined workflow for creating and maintaining safety arguments. Currently these process are encoded in BPMN2 [4] (Business Process Modeling Notation 2) and run on the Activiti workflow engine [1]. Team members work as agents within workflows to achieve an orderly, regular, and auditable form of coordinated work. Automated systems such as build tools and report generators work as agents within the processes to maintain a coordinated view of the project.

Process definitions are provided for various different project types and include roles for developers, reviewers, subject matter experts, auditors, system owners, and regulatory agencies. The library of guidelines and processes available is intended to grow as expertise emerges in the community.

An example project might define an iterative system development lifecycle, in which regular updating and coordination of safety arguments – vetted by internal review and external audit – is enforced. This system might have multiple processes to coordinate system reviews, update safety arguments, lock and merge argument changes, and trigger audit events.

In summary, Maven provides a per-user interface for building a safety case. Git source control provides a means to share safety cases between users. Guidelines and regulations provide knowledge and informal coordination. BPMN2 processes running on an Activiti workflow engine provide rigorous process conformance and coordination where appropriate.

V. Safety Case Presentation

The safety case web site that the SCT build system creates includes HTML pages with document-link features, and automatically numbered sections, figures and tables.

A. Format

The safety case web site is formatted to produce pages with a detailed format designed to support easy access to all the elements of the safety case as necessary. The format of the safety case produced by SCT includes:

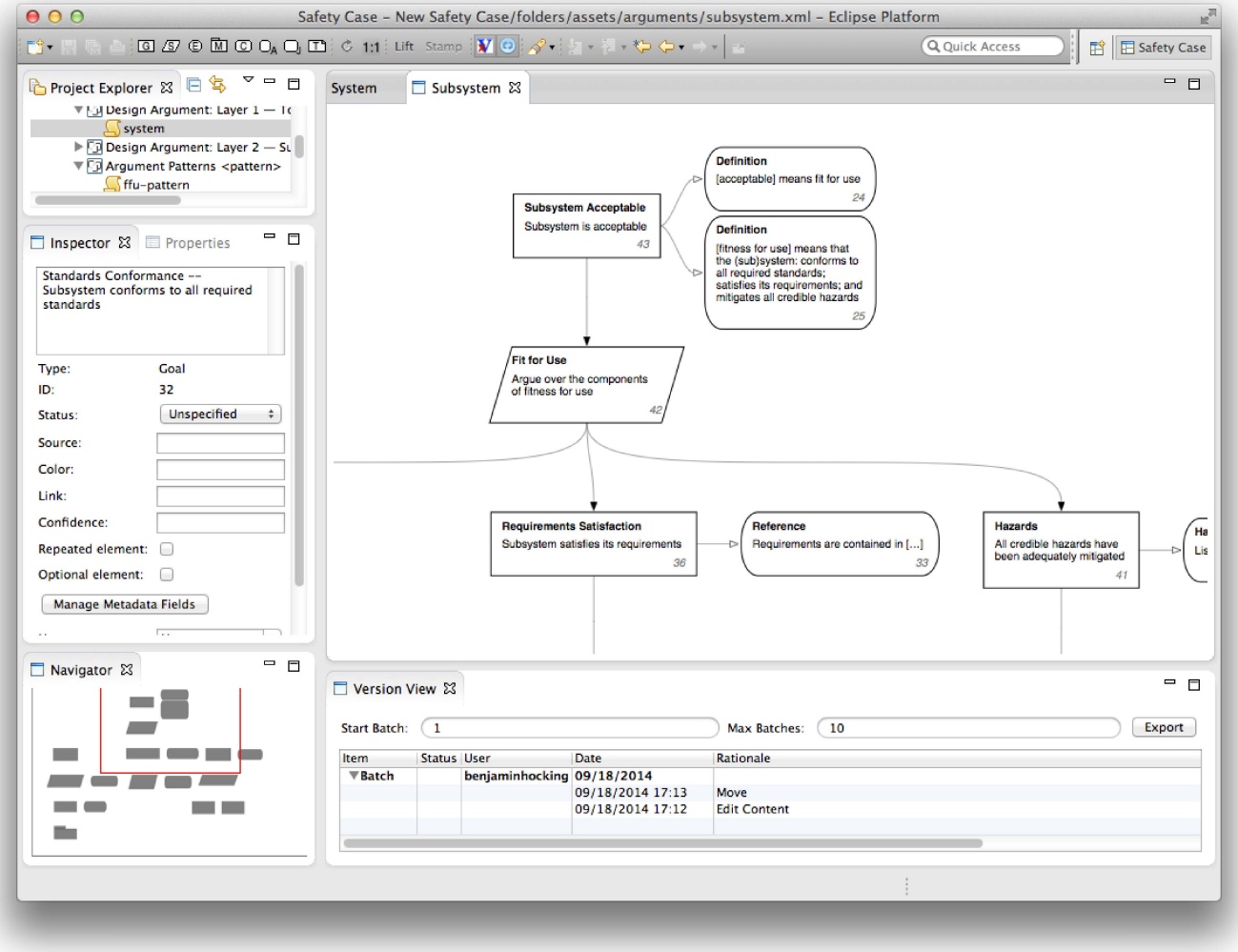

Figure 1 SCT primary interface

- A table of contents.
- A list of figures
- A list of tables,
- Arguments embedded as distinguished figures with specialized built-in functionality.
- Text formatted for ease of review and navigation.
- Hyperlinks to attached documents.

B. *Navigation*

Navigation of the safety case web site is provided by a number of features:

- Hypertext links from assets to other related assets.
- A flyover menu that can be exposed at any time by mousing over a small icon. The flyover menu provides direct access to all of the elements of the safety case from any location in the safety case web site.
- An index of all GSN nodes in all arguments that is automatically synthesized. The index elements are hyperlinked to the associated argument locations. The index is searchable in a variety of ways thereby allowing all of the GSN nodes with a specific characteristic, such as being related to a specific technical point, to be isolated as a group.
- All GSN nodes in all arguments are automatically numbered with left-to-right and top-to-bottom sequence numbers to allow easy reference during discussion. This numbering is temporary, separate from the unique GSN ID mechanism, and purely to facilitate human discussion.

C. *Review:*

An argument *viewer* is integrated into the safety case web site to facilitate argument use and review. Each argument can be viewed either as a figure in the body of the safety case or opened in a separate page. In either case, the viewer provides:

- Argument panning to allow location of the argument on the screen in whatever location the user requires.
- Argument zooming to allow the level of detail that the user requires. Along with display of the argument, a

navigation map is displayed with a thumbnail of the argument to locate the visible portion of the argument.

- Argument focus to facilitate detailed examination of the argument. The focus mechanism displays a selected node along with a user-specified number of levels (e.g., 0-9 from the keyboard) of its descendants. The viewer uses the auto-layout mechanism to display the selected elements in a convenient way for the user.

VI. Toolkit Features

Figure 1 shows a screenshot of the primary SCT interface. Other interfaces are used for managing the lifecycle process mechanisms. The main features of the primary interface are:

- **Editor Panel**. The large panel on the upper right is the *editor panel*. In the screenshot, the GSN editor is shown although other editors are supported. The editor panel supports multiple concurrent displays managed as tabs.
- **Navigator Panel**. The panel on the bottom left is the *navigator panel*. This panel displays the navigation map to support the GSN editor. When an argument is open that is larger than can be accommodated in the editor panel, the entire argument is shown in the navigator panel with the fraction shown in the editor panel delimited with a red rectangle.
- **Inspector Panel**. The panel at the center left is the *inspector panel*. The inspector panel displays all of the fields associated with a selected GSN node and facilitates editing the details of that node.
- **Explorer Panel**. The panel at the upper left is the *explorer panel*. All of the assets that constitute a safety case are accessed from the explorer panel.
- **Version Panel**. The panel at the bottom is the *version panel*. Details of the versions together with the user-supplied rationale for the changes associated with versions are available in this panel.

VII. Conclusion

In this paper, we have presented details of Dependable Computing's Safety Case Toolkit (SCT). The SCT has been under development for about four years and supports several safety cases, including both research and production cases.

The largest system that SCT supports is the safety case for the airspace integration of a large unmanned aircraft system (UAS). This safety case includes a wide variety of evidence from various forms of analysis, simulation, expert judgment, and experimental measurement. The arguments in the safety case are organized into several modules and include hundreds of GSN nodes. The safety case is accessed regularly by dozens of subject-matter experts, technical-area experts, and project managers.

Finally, we note that SCT can also be used for security-critical systems in the development of security cases. Development of experimental security cases using SCT has been successfully undertaken.

Acknowledgments

We thank all of the users of SCT who have contributed comments and suggestions during development of the toolkit.

References

[1] Activiti BPM Platform, http://activiti.org

[2] Apache Maven Project, http://maven.apache.org

[3] Git Version Control System, http://git-scm.com

[4] Object Management Group, Business Process Model And Notation, http://www.omg.org/spec/BPMN/2.0/