Challenges and Benefits of Implementing Hazard Traceability in an Application Lifecycle Management (ALM) Tool

CHIANG Hai Hwee; ST Electronics (Info-Software Systems) Pte Ltd; Singapore
MOK Boon Eang; ST Electronics (Info-Software Systems) Pte Ltd; Singapore
NG Suk Yen; ST Electronics (Info-Software Systems) Pte Ltd; Singapore

Keywords: Requirements Management, Safety Analyses, Requirements Traceability, Hazard Traceability, Tool

Abstract

Traceability of identified hazards to system requirements, design, interfaces and data is required by various system safety engineering standards and certifications such as MIL-STD-882E and RTCA DO-178C. To achieve the traceability requirement, provision for a tracing system is necessary. The tracing system can be a simple spreadsheet or a complicated automatic tracing tool. In this paper, we describe our organization’s common practice of performing requirements traceability and hazard traceability using two different tracing systems – by adopting an application lifecycle management (ALM) tool as a de facto for requirements traceability; and maintaining hazard traceability within hazard records in a textual document coupled with a hazard traceability matrix in a spreadsheet. We present the motivation and efforts to integrate hazard traceability in the ALM tool by leveraging on its existing requirements management features. In particular, we share the challenges of developing and deploying the hazard traceability feature, with an evaluation of its use after the implementation.

Main Body

Introduction

In the various system safety engineering standards and certifications, traceability of identified hazards to system requirements, design, interfaces and data is essential. For example, in the process tasks of system/subsystem/system-of-systems (SoS) hazard analysis, the Military Standard MIL-STD-882E requires that design requirements to preclude hazards are flowed down from top-level specifications to detailed design specifications and test procedures, and that identified SoS hazards are traceable to architecture locations, interfaces and data (US Department of Defense, 2012). RTCA DO-178C, the standard for airborne systems, specifies that trace data should be provided for the bi-directional associations between lifecycle data items contents (such as system requirements, high-level requirements, low-level requirements and test cases) (RTCA, 2011). The Joint Software Systems Safety Engineering Handbook (JSSSEH), which consolidates the best practices for Software System Safety, sums up the software safety requirements traceability as a flow-down from hazards to Software Requirements Specifications (SRS), and from SRS to design, code and test (Joint Software Systems Safety Engineering Workgroup, 2010).

To achieve the traceability requirement, provision for a tracing system is necessary. A tracing system enables the bi-directional traceability of a requirement from its first level inception to its
implementation in a product and the documentation associated with the product. It can be a simple spreadsheet, textual document or a complicated automatic tracing tool (National Aeronautics and Space Administration, 2004).

Through the governance of safety-significant projects within our organization, we observed the common practice of project teams adopting an application lifecycle management (ALM) tool to trace safety requirements through design, test and acceptance; and using a different tracing means such as textual documents to trace safety requirements to its source (that is, hazards and hazard causes). This practice separates traceability into two main paths: (1) the tracing between safety requirements and hazards (to which, for the purpose of discussion in this paper, we will refer as “hazard traceability”); and (2) the flow down of safety requirements to design, implementation, and test (for which we use the term “requirements traceability”).

To understand the reason for such division, we looked into the requirements traceability and hazard traceability in the software development and safety lifecycle processes.

**Traceability Requirements in the Lifecycle Processes**

**requirements traceability.** Requirements traceability is fundamental to good requirements management in software development. The Capability Maturity Model Integration (CMMI), one of the leading process models for software development, includes requirements traceability as one of the specific practices in its requirements management process area (CMMI Product Team, 2010). Besides being “an established tenet in the software engineering community”, requirements traceability is also “essential for assuring that software is safe for use” (Mäder, Jones, Zhang, & Cleland-Huang, 2013).

In order to define requirements traceability in the general software development, we adopt the V-model as a reference process model for the different phases of the development lifecycle. The V-model offers a framework that shows “a transition of requirements down the left-hand side of the V, and ‘a testing relationship going across the V” (Ratcliffe, 2011). It is thus apt to adapt this framework to define the trace path of artefacts at each phase of the Software Development Lifecycle (SDLC), as shown in Figure 1. For the purpose of illustration, the model is simplified with the solid lines showing the requirements traceability of a typical software development project. It may be extended to include hardware development. For example, we can establish traceability of each requirement from customer requirements to system requirements, flowing down to high-level design, low-level requirements and design, and to source code, and vice versa; and between the respective test descriptions and the elements on the left-side of the V-model. The dashed lines are alternative trace links which specific projects may include. The strategy and the granularity of the trace is dependent on the degree of conformance required of the project.
In the domain of system safety engineering, requirements traceability is extended to involve “identifying safety-critical requirements/functions and then tracing them through design, test, acceptance, changes and upgrades, and through retirement” (National Aeronautics and Space Administration, 2004). In the context of system development, safety-critical requirements, also known as safety requirements, are requirements identified from safety analyses to mitigate or resolve any hazards. Hence, safety requirements are part of the requirements in the generic SDLC V-model. In our organization’s practice, requirements designated as safety requirements are each tagged with a “safety-significant” attribute and are, otherwise, managed no differently from other requirements. On this ground of commonality, our project teams familiar with the tracing system for requirements traceability in non-safety-significant projects employ the same system for safety-significant projects.

PTC Integrity, a product which provides a set of application lifecycle management capabilities, is commonly adopted by our project teams. Many other ALM tools such as Rational DOORS are offered by vendors. The ALM tool provides a full suite of features built for requirements management in the system development lifecycle. Artefacts, such as system requirements specifications, software requirements specifications and design specifications, can be easily entered or imported as itemized requirements into the tool using its built-in templates. The benefits of using an ALM tool to perform requirements traceability are evident from the discussions in various papers, for example, in the “Strategic Traceability for Safety-Critical Projects” article written by Patrick Mäder and his co-authors (Mäder, Jones, Zhang, & Cleland-Huang, 2013); and the “Requirements Traceability and the Effect on the System Development Lifecycle (SDLC)” paper by Glenn A. Stout (Stout, 2001).

**hazard traceability.** While it is crucial to maintain traceability between software safety requirements, design requirements, and the actual implementation, hazard traceability is of equal importance (Caffery, et al., 2012). Hazard traceability concerns the traceability between hazards,
hazard causes and safety requirements. Safety requirements are derived from the safety analyses, including the Preliminary Hazard Analyses (PHA) and System/Subsystem Hazard Analyses (SSHA). The safety requirements either mitigate hazards where software is a potential cause, or enable software to be used as a hazard control (National Aeronautics and Space Administration, 2004). Software safety is particularly concerned with the software causal factors linked to individual hazards and ensuring that the mitigation of each causal factor is traced from requirements to design, code, and test (Joint Software Systems Safety Engineering Workgroup, 2010). Figure 2 summarises the trace path required of the hazard traceability.

![Figure 2 — Trace Path of Hazard Traceability](image)

In conjunction with the safety analyses, residual safety risks must be communicated in detail to stakeholders. This can be in the form of hazard records documented in the hazard database record (Joint Software Systems Safety Engineering Workgroup, 2010). An example of the information to be documented in a hazard record is shown in Figure 3, which is adapted and modified from the JSSSEH. Information related to the identified hazards are derived progressively through the stages of the safety analyses. For example, the hazard description, causes and safety requirements are identified during PHA and hazard control design requirements are defined during SSHA.

![Figure 3 — Example of Hazard Record Document](image)

In our organization, project teams record each hazard record in a textual document and refer to the record as Hazard Control Record (HCR). As depicted by the enclosing numbers in Figure 3, the
HCR provides a means to record and trace causal factors to individual hazards ①, and to ensure that mitigation of each causal factor is traced from requirements ② to design ③ and test ④. Multiple HCRs are consolidated in a spreadsheet to produce a Safety Requirements Traceability Matrix, a compilation of respective trace references from hazard record ID, to requirements, design, code and test (Figure 4), and often a contractual deliverable for safety-significant projects.

<table>
<thead>
<tr>
<th>Item #</th>
<th>Hazard Record #</th>
<th>Customer Requirement ID#</th>
<th>System/Software Requirement ID#</th>
<th>Design Requirement ID#</th>
<th>Code Component</th>
<th>Test References</th>
<th>User Manual References</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>H001</td>
<td>Cont #35</td>
<td>SRS #42</td>
<td>SDD #55</td>
<td>(GT)App</td>
<td>Estn</td>
<td>CheckExpiration</td>
</tr>
<tr>
<td>2</td>
<td>H001</td>
<td>Cont #35</td>
<td>SRS #43</td>
<td>SDD #56</td>
<td>(GT)App</td>
<td>Estn</td>
<td>CheckDate</td>
</tr>
<tr>
<td>3</td>
<td>H003</td>
<td>Cont #42</td>
<td>SRS #50</td>
<td>SDD #58</td>
<td>n/a</td>
<td>n/a</td>
<td>User Material Paragraph 3.1</td>
</tr>
</tbody>
</table>

Figure 4 — Example of Safety Requirements Traceability Matrix

Subsequently, safety and design requirements arising from the safety analyses are either added to or updated in the overall system, software and design requirements. This ensures that safety requirements are incorporated, implemented and traced in the overall system in concern. That, in turn, creates the bridge between requirements traceability and hazard traceability.

Motivation for an Integrated Tracing System

integrating hazard traceability in the v-model. At this juncture, we present a modified V-model in Figure 5 to demonstrate the integration between requirements traceability and hazard traceability. The safety requirements in the hazard branch are traced to the low level requirements of the overall system requirements, as represented by the solid line. This establishes the trace from safety requirements to design, code and test inherent in the existing V-model. As with the other dashed lines in the V-model, the dashed line between system requirements and safety requirements is an alternative trace link that specific projects may include. The modified V-model serves to communicate the otherwise loosely defined integration of traceability between the safety lifecycle and the overall development lifecycle processes.
Figure 5 — Modified V-Model with Integration of Hazard Traceability

**motivation for an integrated approach.** While our project teams use the ALM tool to achieve requirements traceability compliance, they perform hazard traceability separately, usually as a “by-product” of keeping record of hazard-related information. The latter is attributed to the role and evidences required of safety engineers in the safety lifecycle. In most instances, safety engineers will perform the hazard analysis process in conjunction with the software development, software test, and Independent Verification and Validation (IV&V) team(s) (US Department of Defense, 2012). At each stage of the hazard analysis process, the safety engineers are responsible for maintaining evidence (in the HCR, for our case) that hazards, causes and mitigations have been identified, implemented, and verified. Very often, unlike the artefacts involved in requirements traceability, the hazard-related information and required evidences are not easily captured by the plain vanilla ALM tool without much customization.

Despite that, it is possible to trace safety requirements to their associated hazards and hazard causes, and between safety requirements, design, code and test using different tracing systems. This task is often tedious and prone to inconsistencies and errors. Within the requirements traceability, although requirements designated as safety requirements are tagged as “safety-significant”, there is the lack of a mechanism to ensure that all safety requirements identified from the hazard analyses are added to, and eventually, tagged and traced in the overall requirements. Conversely, it is labourious to track requirements marked as safety-significant to their respective hazards and hazard causes kept in a separate system. Yet, this tracking is necessary to understand the impact on the system of changing requirements or software elements.

The flaws that entail the fragmented tracing approach can be overcome by leveraging on the features of the ALM tool, which “provides features for establishing, maintaining, and navigating trace links and has the ability to display trace information in formats such as matrices or trace slices” (Mäder, Jones, Zhang, & Cleland-Huang, 2013). We thus embarked on a project to integrate hazard traceability in the tool.
Integrating Hazard Traceability in the ALM Tool

**team composition.** A working team comprising three safety practitioners from the organization’s system/software safety office and one administrator for the ALM tool was formed to work on the project on a part-time basis. The safety practitioners have been involved in governing and providing consultancy, in the aspects of safety processes, to project teams developing safety-significant systems. The administrator of the ALM tool supports the organization’s projects in the set-up of the tool environment. The role of the safety practitioners in the working team included specification of the requirements and design, testing, and coordinating with development project teams for pilot runs. The administrator of the ALM tool played the key role of implementing the design in the tool.

**requirements and design.** The modified V-model with integration of hazard traceability and the HCR textual document were used to facilitate communication of requirements within the working team. The information on the HCR was re-organized into logical segments of hazard, hazard causes and safety requirements. A data dictionary was created for each of the logical segments, snapshots of which are shown in Figure 6. Most fields in the data dictionary were transferred from the HCR template. Various definitions were specified for the fields, including the field types (such as date or short text), allowable list of values (such as I to IV for the initial severity field), and derived values (such as the mishap risk index field which is derived from mishap severity and probability), et al. During the design phase, prototypes of screens were created for the working team to visualize the data entry and the flow between screens.

**development and implementation.** Following the design phase, the prototyped screens were further developed and report templates created. Customized requirement types and trace relationships for hazards, hazard causes and safety requirements were defined. A trace relationship...
was also configured between safety requirements in the hazard branch and the requirements in the existing model of the overall system, thus completing the integration of hazard traceability in the tool.

An important consideration for the design, and hence implementation, was to simplify the steps for the end user. One such example is the automatic creation of traced links between hazard causes and the associated safety requirements. Figure 7 shows the data-entry screen of a hazard cause. The tool is programmed to construct trace links automatically as safety requirements are added for the hazard cause. At the same time, it allows the end user to establish traceability between the safety requirements and associated requirements in the overall system.

![Figure 7 — Establishing Trace Links between Hazard Cause, Safety and System Requirements](image)

**pilot and parallel runs.** One project was selected for the pilot run and another for the parallel run. The first project was in the testing phase and had a different tracing system from the ALM tool. The second project was in the development phase with an existing database of requirements and traceabilities created in the ALM tool. Hence, separate deployments of the new features were performed for: (1) a fresh set-up of the tool with no previous data in the tool’s database, and (2) the tool that was “in-use” with records in the database. Both the pilot and parallel runs provided a platform for the administrator of the ALM tool to perform deployment for each database scenario and a feedback loop to improve the implementation.

**Evaluation of the Implementation**

**benefits.** Integrating hazard traceability in the ALM tool enables hazards to be managed centrally with the overall system requirements repository where every member who has access to the repository is able to search, in a consistent manner as other requirements, and see the on-going safety analyses that he/she may be concerned with.
The benefits inherent in the requirements traceability feature of the tool apply to hazard traceability as well. For example, built-in tracing in the tool provides a mechanism to assign a unique identity to each hazard traceability item like the rest of the requirements traceability items. New traceability items can be added, or existing items deleted, without affecting the unique identity of other items. There is also the built-in capability of trace queries for completeness checks, such as flagging hazards without mitigating requirements, or safety requirements not traced to any requirement specifications. Conversely, when the end user establishes a trace link between a safety requirement and software requirement that has been traced to design, code and test, the hierarchy of the full traced path can be retrieved.

As a result, efficiency is improved in the management of hazards and their traceability, through a seamless flow between data entry screens and trace creation for hazards, hazard causes and safety requirements; and in the generation of queries and reports in various formats and dimensions.

**challenges.** During the implementation, we were met with two main challenges. One emerged from the working team’s limited technical knowledge of the tool, the second being the deployment to a project using a different tool version from the standard baselined version.

Firstly, the working team aimed to deliver the hazard traceability with minimal “tweaking” of the ALM tool’s built-in features so that projects benefit from the suite of supporting features, such as navigating the trace links and generating queries, with little technical and maintenance support. The working team, however, did not have a good knowledge of the internal workings and technicalities of the tool. This resulted in substantial “trials and errors” and re-work, and in some cases, foregoing of some requirements and design. One such occasion was the desire to reproduce the exact layout of the manual textual HCR with its traceability references using the tool’s integrated MS Word document templates, only for the working team to discover later in the development process that much workaround and scripting had to be done. It was after seeking consultation with the tool’s manufacturer’s support team that the working team replaced the use of the MS Word document templates with the built-in report generator in HTML format, which was less ideal in its presentation but required little scripting effort.

Secondly, the project chosen for the parallel run had its database established with requirements and traceability data, and had used a different tool version from the standard baselined version. As a result, the configuration scripts were tailored specifically for the version that the project used. To avoid the unforeseen effects of running the scripts directly on the project’s production server, the working team made an exact duplicate of the original server on a spare server of similar capacity. The production server was then frozen from further updates until the configured environment on the spare server was verified correct and copied back to the live server. Much effort was expended to test that specific version of configuration, under the limited timeframe for the live project to delay updates on its server.

**recommendations.** From this deployment experience, we recommend the integration of hazard traceability to projects at the stage in which a fresh database can be set up on the standard baselined version of the ALM tool. The configuration scripts, once completed and tested, are standard for projects of similar nature.

For on-going projects in which databases have been loaded with existing data, and particularly,
which use a different version of a tool from the standard baselined version, careful planning for deployment is necessary. The extensive deployment effort calls for consideration to integrate hazard traceability for such projects. Factors to consider include the current phase of the project and the number of hazards and safety requirements involved. If the project is in its later phases of activity, or the number of hazards and safety requirements involved is small, the value gained from the integration may not be significant.

As for projects that are required, possibility by contract, to use a different ALM tool such as Rational DOORS, the implementation team is able to save the effort of specifying requirements and design, and commence from the prototyping of screens to final deployment. The decision to implement hazard traceability on a different tool very much depends on the implementation team’s technical and functional knowledge of the tool, and the subsequent number of projects expected to adopt the tool.

Conclusions

Integrating hazard traceability in the same ALM tool as requirements traceability enables hazards and their traceability to be managed centrally and more efficiently in the overall requirements repository. Various factors contribute to the successful implementation of the integrated traceability, including the working team’s technical knowledge of the tool and the allocation of projects’ resources for a collaborative deployment exercise.

The working team recognizes that there are areas with potential for improvement. One such area is to extend the hazard traceability branch to include procedures and training adopted as mitigation measures but which do not fit in the current V-model of traceable artefacts. Further research of the tool’s capability is also essential to enhance the solution, such as to produce more professional-looking reports without much programming tweaks. Given more time and research, deployment effort should be reduced to encourage more projects to adopt the integrated features.

Despite having all these in place, tracing hazards and requirements in an integrated tracing system can only prove effective if all project members exercise due diligence in keeping the repository of requirements and traceability data up-to-date.

References


CHIANG Hai Hwee, Senior System Specialist, 3 Ang Mo Kio Electronics Park Road, #06-01, Singapore 567714, telephone – +65 6679-8959, email – chiang.haíhwee@stengg.com.

Hai Hwee obtained her B.Sc (Computer and Information Systems) from the National University of Singapore. She has over 25 years of IT experience as a system analyst and software developer. She joined the System Safety Office of ST Electronics (Info-Software Systems) in May 2015. Prior to this, she has implemented large-scale systems in the defense and enterprise solutions industry. She is currently working on projects in the defense domain as a safety manager and software safety engineer.

MOK Boon Eang, Senior Manager, 3 Ang Mo Kio Electronics Park Road, #06-01, Singapore 567714, telephone – +65 6679-8957, email – mokbe@stengg.com.

Boon Eang received her B.Sc and M.Sc (Management of Technology) from the National University of Singapore. She has more than 30 years of experience in software engineering and management in many large scale systems such as Air Command and Control Systems, Air Traffic Control System, Port Automated Guided Vehicle Pilot System, and Fire Services Location and Management of Emergency System. She started practicing system software safety since 2004 and has pioneered the establishment of the System Safety Management framework and processes for her company. She leads the System Software Safety governance and process improvements in her company. In addition, she drives the system software safety capability buildup for her company through various projects’ involvement, training programs, sharing sessions and the Singapore Chapter’s activities.

Boon Eang also helps promote system safety through supporting as the System Safety Lead for ST Engineering Electronics group of companies, supporting as an executive committee member of the International System Safety Society Singapore Chapter, supporting as an organizing committee member for the 2008 International System Safety Regional Conference, contributing software safety materials for the Institution of Engineers Singapore (IES) SE manual, training and certification in 2009, and co-authoring a paper and presented it at the 29th International System Safety Conference in 2011.

NG Suk Yen, System Consultant, 3 Ang Mo Kio Electronics Park Road, #06-01, Singapore 567714, telephone – +65 6679-8958, email – ngsy@stengg.com.

Suk Yen obtained her B.Eng (Mech Engineering) from the National University of Singapore and M.Technology (Knowledge Engineering) from the Institute of System Science in Singapore. She has started as Operations Research Officer before transferring to the IT Division during the Y2K downsizing exercise. Since then she has accumulated more than 10 years of software engineering experience, which ranged from supporting 24X7 operational systems, to delivering customized solutions for overseas customers and to managing an independent test team for a large-scale mainframe migration project. At ST
Electronics (Info-Software Systems) Pte. Ltd., she implemented the software safety processes and strives to promote the safety culture by organizing and conducting regular safety training programs.