A Knowledge Integration Approach for Safety-Critical Software Development and Operation Based on the Method Architecture

Shuichiro Yamamoto

A Knowledge Integration approach for Safety-Critical Software Development and Operation based on the Method Architecture

Shuichiro Yamamoto

Nagoya University
syamamoto@acm.org

Abstract. It is necessary to integrate practical software development and operation body of knowledge to deploy development and operation methods for assuring safety. In this paper, an approach based on the method architecture is proposed to develop a Knowledge integration method for describing various software related bodies of knowledge and the safety case for assuring software life cycle and operation processes.

1 Introduction

Information technology (IT) systems have a profound effect on our modern society. In order to assure the safety of the software used in these systems, it is not sufficient to simply confirm the safety of the software that is developed and operated: we also need software development and operation processes that allow the safety of the processes themselves to be verified. Such development and operation processes may be employed in actual software development and operation projects, it is crucial that they are integrated with the development and operation bodies of knowledge already being used by software developers and operators. We, therefore, propose an approach to assure safety throughout all stages of software development and operation by specifying a method for integrating the safety case[1, 2, 3]—a technique used to assure safety both in software and its development and operation processes—with multiple bodies of knowledge based on the method architecture[4], which can be used to describe a variety of practices in a uniform manner.

2 Background

There are high expectations for the safety case in the assurance of system safety in fields such as aerospace, medical devices, and automobiles. In order to promote and support the adoption of this technique in the software development and operation workplace in Japan, the authors established the D-Case Validation & Evaluation Study Session (http://www.dcase.jp) in September 2012. On the international front, meanwhile,

we have participated in efforts aimed at proposing the Assured Architecture Development Method (AADM)[6] to the international standards organization The Open Group (TOG) as a technique for highly safe development of architectures in relation to TOGAF[5]—the group's enterprise architecture framework. As a result of this work, *Open Dependability through Assuredness* (O-DA) became the first standard originating in Japan to be adopted by TOG in July 2013. However, this standard is still quite new, and the safety case approach has not yet made sufficient inroads into the development and operation workplace.

Our experience in the promotion and standardization of the safety case has shown the following to be the main factors behind slow progress in the pickup of this technique at home and overseas:

(1) Software developers and operators, as well as managers at IT and user companies, have no clearly defined objectives for the introduction of the safety case due to insufficient awareness of the approach itself;
(2) The safety case has not yet been sufficiently integrated with bodies of knowledge currently being used by software developers and operators, such as SoftWare Engineering Body Of Knowledge (SWEBOK)[7], Project Management Body Of Knowledge (PMBOK)[8], Capability Maturity Model Integration (CMMI)[9], Business Analysis Body Of Knowledge (BABOK)[10], Requirements Engineering Body Of Knowledge (REBOK)[11], IT Infrastructure Library (ITIL)[12,13], and SQuARE[14]; and
(3) As a result of the above, no clearly defined methods for introduction of the safety case into development and operation processes have been established, and therefore, organization capabilities for effective application of this approach have yet to be realized.

The application of the safety case to safety-critical systems is obligatory in the United States and Europe, and in much the same way as the O-DA standard has been accepted, the safety case is now recognized as an effective and important tool for building consensus between stakeholders such as the client, the software developer, the operator, and supervisory authorities. However, O-DA has just been integrated with TOGAF, and this standard has yet to see full-fledged application by developers and operators; in addition, the safety case has not been sufficiently integrated with development and operation bodies of knowledge other than TOGAF. Other impediments to introduction into the workplace also exist—for example, the relationships between the safety cases for development and operation process and those for the software deliverables have yet to be clearly defined, and no specific methods have been developed for integrating the safety case with conventional approaches such as Fault Tree Analysis (FTA), Failure Mode and Effect Analysis (FMEA), and Hazard and Operability (HAZOP). As a result, there has been limited crossover between knowledge bases for development and operation and safety-case development knowledge, and practices for analysis and evaluation of the safety of system development and operation processes have been inadequate.

Overseas research aimed at reshaping software development practices in a theoretical fashion has led to the proposal of Software Engineering Method and Theory (SEMAT)[15] based on the method architecture, and a Japan subdivision has been set

up by this research project's participants. In its current form, however, SEMAT does not incorporate the safety case or any other operation or project-management knowledge bases. The applicability of the safety case must be enhanced by leveraging a method-architecture perspective in order that the compound bodies of knowledge described above can be put to effective use.

The team leaders and participants in this research project are working to promote the safety case and SEMAT in Japan so that their benefits may be felt in the software development workplace. While certain corporations do appear to be taking steps towards this end, little progress has been made in this country in terms of the integration and advancement of these techniques.

We propose a comprehensive approach to assure safety in all stages of the software development and operation life cycle by combining multiple bodies of knowledge with the safety case based on the concept of the SEMAT method architecture. Furthermore, in consideration of the findings of this research project's team leaders and participants in relation to patterns, editors, and other elemental safety-case techniques, introduction training activities, knowledge engineering techniques, and the reuse of development practices, we also propose an approach to efficiently assure software safety throughout the entirety of the software development and operation life cycle on the basis of the method architecture and plan to verify its benefit through case studies.

The aim of this wide-ranging, advanced research is to facilitate highly safe development and operation of software based on the concept of the method architecture by proposing an approach whereby (1) a range of bodies of knowledge are integrated with the safety case and (2) multiple knowledge bases are efficiently combined and applied at each stage of the software life cycle.

3 Related work

This section provides a description of the method architecture, the safety case, and software-related bodies of knowledge.

3.1 Method Architecture[4]

In the method architecture, multiple practices can be combined to define a method. These practices are described using both the Essence Kernel—a fundamental practice—and the Essence Language. In this way, a practice can be safely combined with many others in order to create super-ordinate methods. The fundamental concept of the method architecture is as follows.

Method: A method comprises multiple practices. Configured from plans and results, methods are dynamic descriptions that support the daily work of developers by describing not only what is expected, but what is actually done.

Practice: A practice is a repeatable approach for achieving a specified objective. It provides a systematic and verifiable procedure for

addressing a specific aspect of a work task. Each practice can be a component element of multiple methods.

Essence Kernel: This kernel constitutes the essential elements for software development techniques. It can be used to define kernels for other domains.

Kernel Language: This domain-specific language is used to define methods, practices, and kernels.

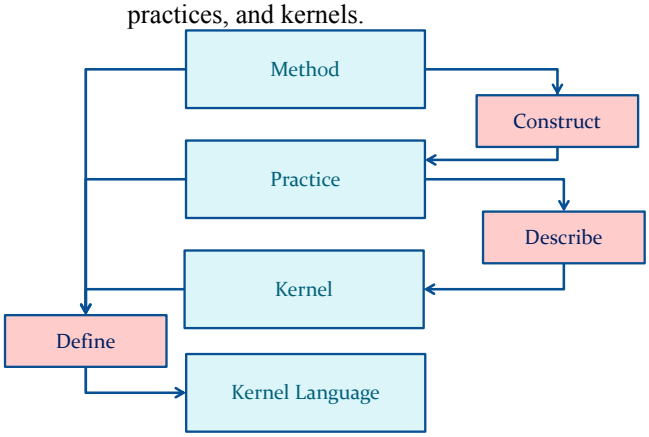

Figure 1: The Method Architecture

3.2 The Safety Case

The component elements of the safety case are claim, strategy, context, evidence, and undeveloped nodes[1][2]. These nodes can be connected to one another in two different ways—namely, by an arrow with a solid head, which is used for claim, strategy, and evidence association, or an arrow with an empty head, which associates a context with a claim or strategy. A typical example of a safety case in Goal Structuring Notation (GSN) is shown in Figure 2.

Claim: A rectangular claim node defines a required property of the system, and it can be decomposed into sub-claims and strategies.

Strategy: A strategy node takes the form of a parallelogram describing an argument required to support the claim. Each strategy can be decomposed into sub-claims or other strategies.

Context: Displayed as a rounded rectangle, a context node provides external information required for correctly interpreting the associated claim or strategy.

Evidence: These nodes provide support for a sub-claim or strategy.

Undeveloped: Shown using a hollow diamond, undeveloped nodes indicate that the corresponding sub-claim or strategy has not been argued.

The authors are working to promote establishment of the O-DA standard within TOGAF[6]. AADM—the O-DA method for assured architecture development—proposes that the following must be established: (1) a management technique for evidence documents and safety cases using an architecture repository; (2) a consensus building technique reflecting claim priorities; (3) a technique for defining the scope of the safety case; (4) a technique for defining quantitative evaluation scales for claims; (5) an evaluation technique for safety-case development capabilities; (6) a safety-case review technique; (7) a technique for combining safety cases; (8) techniques for developing safety cases for the development and operation processes; (9) a technique for combining safety cases with failure-analysis and risk-management techniques; and (10) a tracking technique for safety-case claims and system requirements. In order to achieve this, the authors are studying the formulation of safety case patterns as well as methods for developing safety cases for the development and operation processes [16-20].

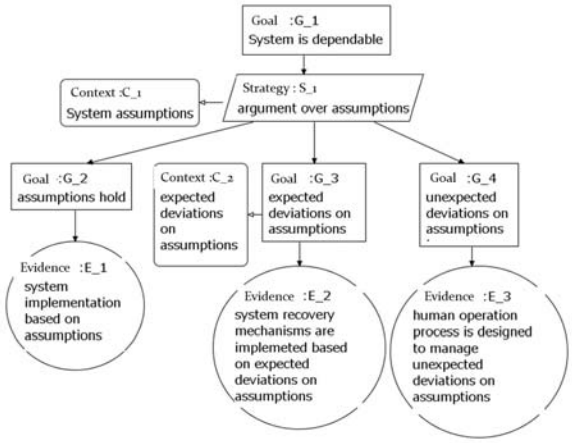

Figure 2: Typical Safety Case

Sections 3.3 through 3.10 provide an overview of software-related bodies of knowledge; Section 3.11 presents the results of a comparison thereof focusing on commonality and identifies research tasks to be undertaken going forward.

3.3 SWEBOK[7]

For each knowledge area, the *Guide to the Software Engineering Body of Knowledge* (SWEBOK® Guide V3.0) provides (1) an introduction, (2) component elements (topics), (3) a correspondence table of topics reference documents, (4), recommended reference documents, and (5) a list of related documents.

SWEBOK knowledge areas are decomposed into two or three hierarchical levels comprising topics. This hierarchical decomposition is configured so as not to include any dependency on specific fields of application, business uses, development techniques, and the like. However, SWEBOK recommends that necessary information be

directly acquired from reference documents, and therefore, it does not itself describe knowledge in detail.

The SWEBOK knowledge fields are: (1) software requirements, (2) software design, (3) software construction, (4) software testing, (5) software maintenance, (6) software configuration management, (7) software engineering management, (8) software engineering process, (9) software engineering tools and methods, and (10) software quality.

3.4 PMBOK[8]

PMBOK recognizes that processes can fall into different knowledge areas and different process groups. The PMBOK knowledge areas are: (1) project integration management, (2) project scope management, (3) project time management, (4) project cost management, (5) project quality management, (6) project human resource management, (7) project communications management, (8) project risk management, and (9) project procurement management. Meanwhile, the process groups are (1) initiating, (2) planning, (3) executing, (4) monitoring and controlling, and (5) closing.

3.5 CMMI[9]

CMMI models are systematized collections of knowledge that help organizations improve their development processes. In accordance with the CMMI for Development, Version 1.3 model, process areas are described using purpose statements, introductory notes, related process areas, specific goals, practice summaries, example work products, generic goals, and generic practices.

3.6 BABOK[10]

BABOK 2.0 systematically describes business analysis knowledge in the form of three hierarchical levels—(1) knowledge areas, (2) tasks, and (3) techniques. Knowledge areas are described in the form of (1) a knowledge area definition, (2) related tasks, (3) related techniques, (4) inputs to the knowledge area, and (5) outputs from the knowledge area. Meanwhile, each task is presented in the form of (1) its purpose, (2) a description of its content, (3) relevant stakeholders, (4) inputs, (5) outputs, (6) task elements, and (7) techniques for performing the task.

3.7 REBOK[11]

REBOK is a body of knowledge intended to support practical requirements engineering in regard to requirements negotiated by users and vendors. Its principal features are as follows:

(1) It is common to both users and vendors;
(2) It organizes standards and knowledge areas that may need to be acquired not only by requirements analysts, but also by end users, managers, and all other stakeholders participating in requirements engineering;

(3) It reflects the scope of business requirements, system requirements, and software requirements; and
(4) It covers techniques common to requirements engineering for enterprise systems and embedded systems (although domain-specific knowledge is defined separately).

3.8 **ITIL**[12,13]

ITIL is a body of knowledge for best practices in service management that allows organizations to provide customers with services that are safe, highly reliable, and meet their objectives, and to also become trusted providers. As a process-based framework covering the entire service life cycle, it comprises five individual processes—namely, service strategy, service design, service transition, service operation, and continual service improvement.

3.9 **TOGAF**[5,6]

TOGAF Version 9 comprises seven parts. Part 1 provides an explanation of the main concept and definitions of terms used. The Architecture Development Method (ADM), which is a step-by-step approach to developing enterprise architectures is described in Part 2 in terms of purposes, objectives, processes, inputs, and outputs. Part 3 provides guidelines and describes techniques for application of the ADM. The TOGAF architecture content framework, which includes a metamodel, architecture building blocks for re-use, and the deliverables of the various ADM phases, is described in Part 4. Part 5 describes the Enterprise Continuum—a categorization system for storing the deliverables of enterprise architecture activities—as well as related tools. The Enterprise Continuum can be seen as a mechanism for categorizing, associating, and storing all information produced in relation to an enterprise architecture. Part 6 provides a description of architectural reference models primarily in terms of the TOGAF Foundation Architecture and the Integrated Information Infrastructure Reference Model. Using the Enterprise Continuum and these reference models, the state of practical implementation of the enterprise architecture can be elicited from business capabilities and the current state of business can be presented with respect to the business vision. Finally, Part 7 describes the organization, processes, skills, roles, and responsibilities required to manage enterprise-architecture activities in terms of the Architecture Capability Framework.

3.10 **SQuaRE**[14]

Software Product Quality Requirements and Evaluation (SQuaRE) is a new standard for evaluating the quality requirements of software products. The SQuaRE standard covers scope, conformance, normative references, terms and definitions, a software quality requirement framework, and requirements for quality requirements. The software quality requirement framework is described in terms of purpose, software and

systems, stakeholders and stakeholder requirements, software properties, a software quality measurement model, software quality requirements, system requirements categorization, and a quality requirements life cycle model. Requirements for quality requirements take the form of general requirements, stakeholder requirements, and software requirements.

3.11 Comparison of Bodies of Knowledge

Table 1 shows the results of analysis of the common content of software bodies of knowledge. From this, we can see that certain content is shared by multiple bodies of knowledge, and therefore, that they should be integrated in an appropriate fashion. As stated in this paper, therefore, a technique based on the method architecture for describing bodies of knowledge in a cross-body manner and safely and efficiently integrating these descriptions is required.

4 Research issues

We propose an approach to support efficient and high-quality development and operation of safety-critical software by making it easier for developers and operators to introduce the safety case, and furthermore, we plan to develop the associated support tools. The following three research tasks must be undertaken in order that the safety case may be implemented in all stages of the software life cycle from development through operation.

4.1 Enhancement of Safety in Software Development & Operation Processes with the Safety Case

Based on specific case studies of software development and operation, impediments to practical implementation of the safety case must be identified and effective solutions proposed. Using the O-DA AADM, for example, we must confirm that safety cases can be developed and that critical safety can be analyzed and assured for all data processed by information systems, beginning with the business vision; for the execution of application functionality and the information technologies utilized; and for system deployment and operation scenarios. Further, we also propose a safety extension similar to the O-DA for bodies of knowledge other than TOGAF, such as BABOK and ITIL.

In order to assure safety, we must confirm that safety countermeasures for mitigating system risk to the greatest possible extent have been implemented with software. We thus propose (1) an approach for enumerating risk factors related to system requirements and system design and reviewing the completeness of the safety case, and (2) an approach for confirming that safety requirements and designs are free of omissions by preparing an ontology and case-study basis for system failure.

4.2 Reconfiguration of Bodies of Knowledge with Method Architecture

We propose an approach whereby the above-described safety case and method for enhancing the safety of bodies of knowledge currently used in the workplace are integrated in a manner that spans multiple bodies of knowledge in order to facilitate smooth implementation by developers and operators. More specifically, we propose an approach for confirming and assuring the validity of highly safe software that—based on the method architecture—can systematically integrate a software-engineering body of knowledge, a requirements-engineering body of knowledge, a project-management body of knowledge, an operation body of knowledge, and so forth. In particular, this approach would allow for the assurance of safety-related intentions by software and its development and operation processes to be objectively explained and confirmed based on evidence from the safety case. Research into this approach with thus focus on (1) cross-body integration based on objective evidence common to multiple bodies of knowledge, and (2) verification of the effectiveness thereof on the basis of case studies.

4.3 Support Tool Development

We plan to design and prototype a tool that can provide highly effective support for this paper's proposed development and operation approach for safety-critical software based on the method architecture. Editors[21] and other tools for the development of safety cases allow safety-case patterns and failure patterns based on application case studies to be managed in repositories and reused. Furthermore, we hope to realize support functionality for analyzing safety confidence levels and the completeness of arguments. We will also develop ontologies for eliminating inconsistencies in terminology and other knowledge arising from the combination of different bodies of knowledge and the reuse of safety cases. These repositories and ontologies will contain guidelines and training materials for practical application of these bodies of knowledge with higher levels of safety due to implementation of the safety case, and they will also prove useful in training and promotion related to the development and operation approach employing the method architecture.

5 Considerations

5.1 Practice and Theory

In proposing an approach based on the method architecture for efficiently integrating the safety case with the various bodies of knowledge put to practical use by developers and operators of safety-critical software, this research project is highly practical in nature. Meanwhile, it also has an academic aspect in proposing a theory for cross-body

integration of compound bodies of knowledge on software development and operation with the safety case.

5.2 Development Process for Proposed Approaches

We aim to contribute to greater levels of safety and security in software providing social infrastructure by collecting and reusing valuable knowledge through the study of case studies on actual software development and operation and also by publicizing and promoting the above-described approaches in the form of seminars, study sessions, and so forth.

5.3 Enhancing the Safety of Software Bodies of Knowledge

In terms of the practical development and operation of safety-critical software both in Japan and overseas, we also aim to contribute to higher levels of safety in these processes not only by enhancing the safety of the software itself based on principles and targets but by also systematically reconfiguring software development and operation knowledge bases according to the method architecture concept.

6 Summary and Future Issues

In this paper, we introduced efforts aimed at enhancing the safety of software-related bodies of knowledge utilized by the developers and operators of safety-critical software and efficiently integrating these bodies of knowledge using the method architecture and the safety case in order that this type of software may be developed and operated in a more practical fashion. We have examined eight specific bodies of knowledge in this regard, but in order to generalize our research, more must be evaluated. In regard to testing, for example, a method has been established for confirming sufficiency using the safety case (or, more precisely, the assurance case)[17], and therefore, it is necessary to integrate the bodies of knowledge considered in this research using the method architecture.

Figure 3 shows an overview of our proposal. Going forward, we plan to conduct research focusing on the tasks identified in Section 4. In terms of the technical knowledge contained in the various different bodies of knowledge, meanwhile, we will examine (1) how it can be integrated and (2) how a high level of safety can be achieved by means of the safety case.

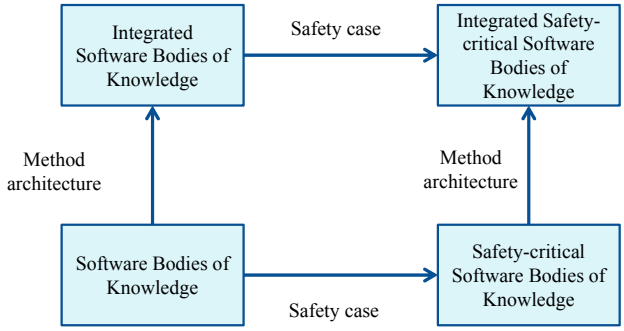

Figure 3: Integration of Bodies of Knowledge for Safety-Critical Software

References

1. Kelly, T. P., "A Six-Step Method for the Development of Goal Structures", York Software Engineering, 1997
2. Kelly, T. P., "Arguing Safety, a Systematic Approach to Managing Safety Cases". PhD Thesis, Department of Computer Science, University of York, 1998
3. Bloomfield, P., Bishop, P., 'Safety and Assurance Cases: Past, Present and Possible Future – an Adelard Perspective", 2010
4. OMG, "Essence – Kernel and Language for Software Engineering Methods", ad/2013-02-01, http://www.omg.org/spec/Essence/1.0, 2013
5. The Open Group, TOGAF V.9 A Pocket Guide, 2008
6. The Open Group, "The Open Group Real-Time & Embedded Systems Forum, Dependability through Assuredness™ Standard (O-DA)", 2013
7. Guide to the Software Engineering Body of Knowledge (SWEBOK V3), http://www.computer.org/portal/web/swebok/home
8. "PMBOK Guide", http://www.pmi.org/
9. CMMI, CMU/SEI-2010-TR-033, 2010
10. IIBA, Japan Chapter, "A Guide to the Business Analysis Body of Knowledge", 2009
11. Japan Information Technology Services Industry Association, REBOK Working Group, "Requirements Engineering Body Of Knowledge", http://www.seto.nanzan-u.ac.jp/~amikio/NISE/eng/REBOK/REBOK-APSEC2011-Tutorial-2011-12-05.pdf
12. ITIL, itSMF Japan, http://www.itsmf-japan.org/itil
13. iTSMF, ITIL V3 Foundation Handbook, 2009
14. Boegh, J., "A New Standard for Quality Requirements", IEEE Software, pp. 20-27, January/February, 2008.
15. Jacobson, I., Ng, P-W., McMahon, P. E., Spence, I., Lidman, S., "The Essence of Software Engineering – Applying the SEMAT Kernel", Addison-Wesley Pearson Education, 2013
16. Yamamoto, S., Matsuno, Y., "d* framework: Inter-Dependency Model for Dependability", DSN 2012
17. Yamamoto, S., Matsuno, Y., "A Review Method based on a Matrix Interpretation of GSN", JCKBSE2012
18. Matsuno, Y., Yamamoto, S., "Consensus Building and In-operation Assurance For Service Dependability?", CD-ARES 2012, LNCS 7465, pp. 639-653.

19. Yamamoto, S., Kaneko, T., Tanaka, H., "A Proposal on Security Case based on Common Criteria", Asia ARES 2013
20. Yamamoto, S., Matsuno, Y., "An Evaluation of Argument Patterns to Reduce Pitfalls of Applying Assurance Case", 1st International Workshop on Assurance Cases for Software-intensive Systems (Assure 2013)
21. Matsuno, Y., Yamamoto, S., "An Implementation of GSN Community Standard", 1st International Workshop on Assurance Cases for Software-intensive Systems (Assure 2013)

Table 1: Analysis of Commonality of Software Bodies of Knowledge

	SWEBOK	PMBOK	CMMI	BABOK	REBOK	ITIL	TOGAF	SQuaRE
SWEBOK	Software development	Development projects	Development management	Requirements engineering	Requirements engineering	Design and transition	Solution deployment monitoring	Software quality
PMBOK		Project management	Development management	Requirements management	Risk management	Operation management	EA management	Quality improvement processes
CMMI			Development, procurement, and services	Process improvement	Process improvement	Service maturity	EA maturity	Quality improvement processes
BABOK				Business analysis	Business requirements	Business requirements	Business architecture	Quality assurance
REBOK					Requirements engineering	Operation requirements	Requirements management	Quality requirements
ITIL						Services	Operation monitoring	Service quality
TOGAF							EA	EA quality
SQuaRE								Software quality