

A Thought Experiment on Evolution of Assurance Cases —from a Logical Aspect

Shuji Kinoshita and Yoshiki Kinoshita^(✉)

Kanagawa University, Tsuchiya 2946, Hiratsuka 259-1293, Japan
{shuji,yoshiki}@progsci.info.kanagawa-u.ac.jp

Abstract. A thought experiment on evolution of assurance argument is performed on the basis of an interview with a manufacturer that applied for a certification of conformance of their in-house software life cycle to a safety standard. The working hypothesis of the experiment is that assurance cases help find problems in arguments on software life cycle and improve the life cycle. Based on the result of the thought experiment, questions for further empirical studies are generated and the ontology of relevant information items are analysed.

Keywords: Assurance case · Assurance argument · Software assurance · Software life cycle · Evolution · Formal approach · Thought experiment

1 Introduction

Assurance arguments must change as their target item (product, system, its life cycle or service) changes. The item changes because of corrections or changes in its environment, including the system connected to the item and the relevant regulations.

The authors had an interview with a manufacturer who applied for a certification for conformance of their in-house software life cycle to a safety standard. They struggled to follow update of the standard. In some occasions, the manufacturer had a valid argument for the conformance of their initial software life cycle, but the certification body found it unacceptable and the manufacturer had to modify their life cycle in order to give an argument that the life cycle conforms to the revised standard. The manufacturer recorded their initial software life cycle but did not record their assurance argument on it in such a way that enables tracing the problem.

The authors surmised that a logical support by means of assurance cases could have reduced the struggle after the rejection. This led the authors to set up the working hypothesis that assurance cases help find logical problems in arguments on software life cycle and improve the life cycle and perform a thought experiment that explicates how assurance cases help.

Logical here does not only mean “according to logical inference” but also “according to ontology” (i.e., vocabulary and basic assumptions that frames the arguments). In order to focus on logical aspects of assurance arguments and to abstract away from the irrelevant aspects, a hypothetical safety standard and hypothetical software life cycles are used in this work.

The thought experiment is performed in the following hypothetical setting. It is assumed that a software manufacturer is applying for certification that their own in-house *software life cycle*, named SLC1, conforms to a hypothetical safety standard STD (Fig. 1). Before application, the manufacturer has had an assurance argument that convinced themselves of the conformance and has developed an assurance case A1 that records the conformance argument. For that purpose, there is a need for an interpretation I1 of STD in the context of SLC1, and I1 is also documented in A1.

Unfortunately, the first application for a certification is rejected. The manufacturer has to find the reason of rejection by itself because the certification body does not provide it, as is common in certification. The assurance case A1 is examined and two possible reasons for rejection are found. To remedy these problems, the interpretation of the standard in the context of the life cycle is changed.

The manufacturer revises the interpretation and its in-house software life cycle to obtain I2 and SLC2, and the assurance case A2 is developed to record the relevant assurance argument. The scenario finishes with the success of the second application.

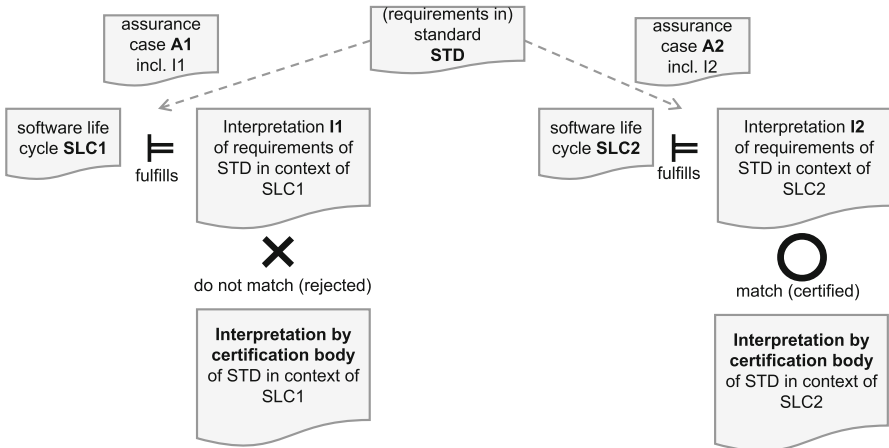


Fig. 1. Information items of the thought experiment

After the thought experiment, three questions are generated out of its result. These questions may serve as a starting point and determine a framework of further studies, which could be empirical work or other thought experiments.

The ontology of relevant information items is also analysed. Information items include standards, descriptions of items (software life cycle, in this work) to conform to the standard, and interpretations of requirements of the standard in the context of the item.

To sum up, the contribution of this work is two folds: generation of questions for further studies and analysis of ontology of relevant information items.

This paper is organized as follows. The thought experiment of assurance argument is presented in Sects. 2–7. A hypothetical safety standard STD is presented in Sect. 2. The standard, the following software life cycles and assurance cases are presented to the extent only enough for the discussion in this paper. Section 3 presents a hypothetical software life cycle SLC1 for in-house use in a manufacturer. Section 4 provides the manufacturer’s interpretation I1 of requirements of STD in the context of SLC1 and assurance argument claiming that SLC1 conforms to the requirements of I1; these are compiled as the assurance case A2. Sections 6 and 7 are about the modified software life cycle and assurance case developed in response to rejection of application for a certification. This ends the presentation of the thought experiment.

Questions generated by this thought experiment and the ontology for relevant information items is discussed in Sect. 8, where relevant to this thought experiment is also analysed. Section 9 concludes with a list of related work.

2 The Hypothetical Safety Standard STD

Our hypothetical safety standard STD specifies requirements on software life cycle processes on top of the general requirements in ISO/IEC/IEEE 12207 [4]. For risk management, it contains the following requirement, among others.

(STD-1) The risk under the intended use of the target software and reasonably foreseeable misuse shall be managed. The result of risk management shall be documented.

In the sequel, we focus on conformance to this requirement.

3 The Software Life Cycle SLC1

The manufacturer has a software life cycle for its in-house use. It contains the following processes, among others. The reader is referred to Fig. 2.

- (SLC1-1) Define and record stakeholder needs and requirements. The output includes the stakeholder needs and requirements, and records the intended use of the software. It also records all misuses that are identified according to the activity SLC1-0 (not shown).
- (SLC1-2) Define safety requirements, which identify the requirements for safety under the intended use and misuse identified in the document. The input to safety requirements definition includes the stakeholder needs and requirements.

(SLC1-3) Manage risk. The activities include the following:

- (SLC1-3)(1) analyse risk of the system satisfying the system/software requirements,
(SLC1-3)(2) evaluate risk analysed above, and
(SLC1-3)(3) if the result of risk evaluation requires, provide means of controlling risk and implement it; this task includes selection of risk control measures, review of new risk introduced by the selected risk control measures, and review of risk increased by the selected risk control measures. The input to risk management includes the system/software requirements among others. All output is recorded in risk management file.

The input to risk management includes the system/software requirements among others. All output is recorded in risk management file.

- (SLC1-4) Define system/software requirements. The input includes stakeholder needs and requirements, safety requirements and risk management file, among others.

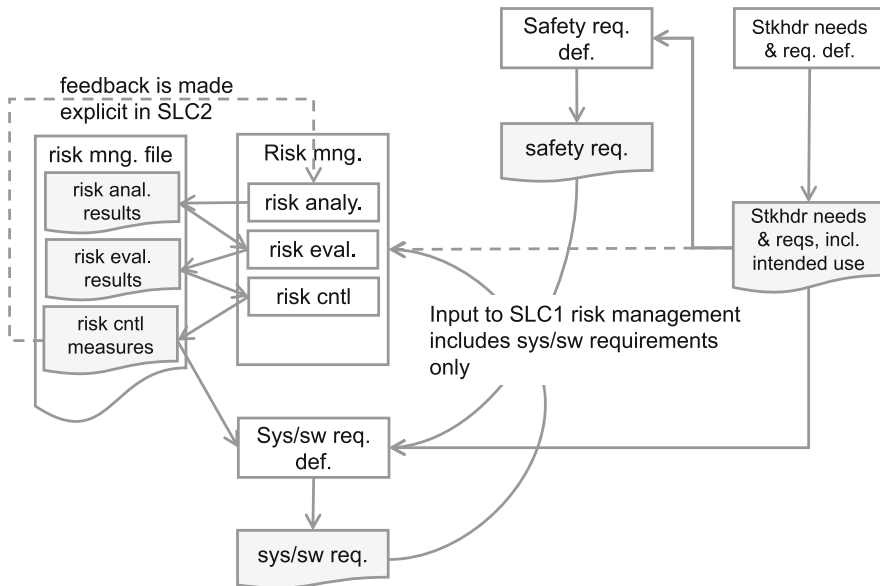


Fig. 2. Processes and their input/output

4 Assurance Case A1 that Claims Conformance of SLC1 to STD

The manufacturer interprets the requirement (STD-1) in the context of SLC1 as (I1-1)–(I1-3); this interpretation is called I1. It then developed an assurance argument that claims the conformance of SLC1 to I1 and documents the whole argument including I1 as the assurance case A1.

The target of the argument is software life cycle here, and is not to be confused with a concrete software. Therefore, for instance, we talk about whether a provision of assurance argument on the software is included in the life cycle, instead of talking about whether some concrete assurance argument on the software is valid and appropriate.

- (A1-1) The following three claims are sufficient for the conformance to (STD-1).
 - (I1-1) The risk of the system satisfying the system/software requirements shall be managed.
 - (I1-2) Risks newly introduced by the selected risk control measures and the risks increased by the selected risk control measures shall be reviewed.
 - (I1-3) The result of risk management shall be documented as the risk management file.
- (A1-2) (I1-1) holds for SLC1 because the risk of the system satisfying the system/software requirements is managed (SLC1-3)(1).
- (A1-3) (I1-2) holds for SLC1 because of (SLC1-3)(3).
- (A1-4) (I1-3) is supported by the last statement of (SLC1-3).

5 Rejection of Certification and Improvement of SLC1

The manufacturer applies for certification of conformance of SLC1 to STD. Unfortunately, it is rejected without any reasons specified, as is often the case for certification. So the manufacturer examines A1 and concludes there can be the following two reasons for rejection.

- (RR-1) (I1-1) may need to be strengthened. The requirement (STD-1) of STD requires to manage the risk under intended use and identified misuse. The requirement (I1-1) is considered appropriate as an interpretation of (STD-1) because the system/software requirements are derived from stakeholder needs and requirements including intended use and identified misuse. However, this rationale is not made explicit and may be insufficient.
- (RR-2) (I1-2) may not be sufficient because then there may be no room for *iterative* realization and evaluation of the risk control measures. The new risk and increased risk found in (SLC1-3)(3) must be managed.

The interpretation I1 is updated in the following way. To resolve (RR-1), the following requirement is added in the new interpretation: to fulfill (STD-1), assurance argument for the claim shall be provided that the risk from each intended use and reasonably foreseeable misuse are managed to a sufficient level. To resolve (RR-2), the interpreted requirement (I1-2) was strengthened so that “risks newly introduced by the selected risk control measures and the risks increased by them” shall not only be reviewed but also be managed.

These changes necessitates the change of the life cycle SLC1 and argument A1. An activity that fulfills the demand for assurance argument is added to the life cycle to fulfill (I1-1). For (I1-2), the risk management process of the life cycle is made iterative so that the result of (SLC1-3)(3) is input to (SLC1-3)(1). The iteration is to be terminated when the new and increased risk is small enough so that it is acceptable as residual risk.

6 Improved Software Life Cycle SLC2

The revised software life cycle SLC2 contains the following processes. The difference from SLC1 is indicated by underline.

- (SLC2-1) Define and record stakeholder needs and requirements. The output includes the stakeholder needs and requirements, and records the intended use of the software. It also records all misuses that are identified according to the activity SLC2-0 (not shown).
- (SLC2-2) Define safety requirements, which identify the requirements for safety under the intended use and misuse identified in the document. The input to safety requirements definition includes the stakeholder needs and requirements.
- (SLC2-3) Manage risk. The activities include the following:
 - (SLC2-3)(1) analyse risk of the system satisfying the stakeholder needs and requirements,
 - (SLC2-3)(2) evaluate risk analysed above, and
 - (SLC2-3)(3) if the result of risk evaluation requires, provide means of controlling risk and implement it; this task includes selection of risk control measures, review of new risk introduced by the selected risk control measures, and review of risk increased by the selected risk control measures. The result of review is then input to SLC2-3(1) to start the risk management iteratively. (This iteration is terminated when the result of risk evaluation does not require risk control measures in (SLC2-3)(3).)

The input to risk management includes the stakeholder needs and requirements and the system/software requirements among others. All output is recorded in risk management file.
- (SLC2-4) Define system/software requirements. The input includes safety requirements and risk management file, among others.
- (SLC2-5) For each use and misuse identified in (SLC2-1), provide an assurance argument for the claim that the risk is managed by (SLC2-3) appropriately.

7 Assurance Case A2 that Claims Conformance of SLC2 to STD

The manufacturer interprets the requirement (STD-1) in the context of SLC2 as (I2-1)–(I2-3); this interpretation is called I2. It then developed an assurance argument that claims the conformance of SLC2 to I2 and document the whole argument including I2 as the assurance case A2. The difference from A1 is indicated by underline.

- (A2-1) The following three claims are sufficient for the conformance to (STD-1).
 - (I2-1) The risk of the system under the intended use and identified reasonably foreseeable misuse of the target software shall be managed. Also, an assurance argument for the claim shall be provided that the risk under each intended use and reasonably foreseeable misuse are managed to the sufficient level.

(I2-2) Risks newly introduced by the selected risk control measures and the risks increased by the selected risk control measures shall be reviewed and managed.

(I2-3) The result of risk management shall be documented as the risk management file.

(A2-2) (I2-1) holds for SLC2 because the risk of the system satisfying stakeholder needs and requirements is managed (SLC2-3), the stakeholder needs and requirements include the identified intended use and reasonably foreseeable misuse (SLC2-2), and for each intended use and foreseen misuse, an assurance argument is provided for the claim that the risk is managed properly.

(A2-3) (I2-2) holds for SLC2 because of (SLC2-3)(3).

(A2-4) (I2-3) is supported by the last statement of (SLC2-3).

The second application of certification that SLC2 conforms to STD with the assurance case A2 was accepted by the certification body.

8 Discussion

8.1 Questions Generated by the Thought Experiment

Some questions arise from the result of the thought experiment.

The first question is

How difficult or easy it would be to find the problems (RR-1) and (RR-2) if assurance case A1 were not developed?

Use of the assurance case A1 enabled finding the problem (RR-1). If assurance cases were not used, the manufacturer would have to recall the detail of their implicit argument for the reexamination after the rejection. It would have been difficult to spot the difference between the risk under intended use and identified misuse and the risk of the system satisfying system/software requirements.

Finding (RR-2) would require similar detail. Here, the difference between only reviewing the risk and reviewing it and starting another risk management cycle is enabled by means of examination of explicit description provided by A1.

The assurance cases clarify the ontology by distinguishing the requirements in the standard, their interpretations and the description of the life cycles. Empirical work or another thought experiments may be conducted in order to investigate whether this makes it easier to find the problems such as (RR-1) and (RR-2).

The second question is

How could formal approach to assurance argument facilitate the manufacturer's work and the certification body's work?

Formal approach to assurance arguments, as proposed in [8] would provide methods to describe ontology formally, automatic methods to check the formal ontology, and methods to experiment on the formal ontology by means of proof checkers or theorem provers. This is similar to the case in validation where the

specification alone is examined but not quite the same as the case in verification where the implementation is examined with respect to the given specification.

In the presented thought experiment, it does not matter whether the assurance argument is given formally or not.

The second question may be specialised to the following third question.

How could formal approach to assurance argument facilitate to handle logical flaws in the argument?

Finding and correcting logical flaws in the argument is not included in the present thought experiment, but there are certainly those reported in authors interview with manufacturer. Formal approach obviously enables finding logical flaws, but the significance of such finding in the context of the overall assurance of total software life cycle should ultimately be examined *in vivo* by empirical study, by thought experiments or by other means.

These questions may serve as a starting point and may provide a framework in which further investigation may be conducted.

8.2 Information Items and Relation Between Them

In this thought experiment, the following kinds of information items appear.

- (1) Top level requirements of interest (the standard STD, in this case).
- (2) Item of interest (software life cycles SLC1 and SLC2).
- (3) Interpretation of the top level requirements in the context of the item (I1 and I2).
- (4) Assurance argument that the item of interest fulfills the interpreted requirements (A1 and A2).

Note that we do need the interpretation that instantiates the requirements in the setting (or in the vocabulary and basic assumptions) of the item of interest because the top level requirements is not usually written in the context of a specific item, i.e., in the vocabulary of the item nor under the basic assumption that should be supposed to be the case for the item.

These information items have interdependency with each other. Items are expected to *conform* to the interpreted requirements, and the interpreted requirements should be related to the top level requirement so that the conformance of any applicable item to the interpreted requirements *implies* that to the top level requirements.¹ Logical and mathematical formulation of these

¹ Here is a more strict but involved explanation. Items and the top level requirements are described in a different language as they are at different level of abstraction. So, fulfillment by an item of the top level requirements means fulfillment by an item of the *interpreted* top level requirements, given an interpretation of top level requirements language to items language. There are in general many such interpretations, such as the manufacturer's and certification body's in our thought experiment. Stakeholders would have their own interpretation under which they are confident that "an item fulfills the requirements" means "an item fulfills the interpreted requirements."

relations is not trivial and theories such as clones in universal algebra and Lawvere theories in category theory seems relevant. Such formulation would work as the theory behind formal approach to formulate these information items in the direction suggested in [8]. Investigation in this direction is left as future work.

In the thought experiment, typified uses of the verbal forms used in requirements and specifications of life cycle processes are of much help. For instance, requirements (top level requirements and their interpretations) are typically stated as “shall be,” and life cycle processes are typically described by imperative mood, such as “Act.” Some of these rules are provided in the manual for standard writing such as [3], and to follow such rules would ease mathematical formulation of the relationship between the information items.

9 Conclusion and Related Work

A thought experiment was performed, with a hypothetical setting that a manufacturer is applying for certification of conformance of their own in-house software life cycle to a hypothetical safety standard. Three questions were generated out of the result of the experiment. Approaches to the questions were suggested but the study of these questions remains as future work. Ontology of information items relevant to this experiment and to assurance arguments on software life cycles in general were also discussed.

There are works on assurance cases and conformance to standards including the following. [1] reports practical experience concerning structuring assurance case in the context of conformance to actual standards. [2] presents an explicit framework of assurance case as required by DO-178C. A similar framework is found in IEC 62853 Open systems dependability, whose Committee Draft for Vote is in the process of approval at the time of preparing this paper.

[6] discusses integration of formal arguments into system refinement in the context of system/software life cycle. It refers to the changes in connection to modularization and encapsulation of the system. [7] argues that assurance case plays an essential role in the change management for items (product including software and service). Specific example of change, however, is not given in these works. Design of system/software life cycle is not considered, either.

The thought experiment in the current work examines the change to the system/software life cycle and assurance argument on it in the context of conformance to a hypothetical standard.

[5] discusses that standards should define the properties of the target item that are expected to accrue from the use of the standard, as well as a rationale that justifies the contents of the standard. This work may be regarded to proceed in the same direction as the analysis of information items in this work.

Acknowledgments. The authors acknowledge Makoto Takeyama’s thoughtful comments on the draft of this paper. Koji Okuno coordinated the authors’ contact with Nihon Kodensho Corp. that led to this work. The authors thank Kazuo Oosone, Masato Tanaka and Yuichi Kurabe for sharing their experience as software engineering experts in industry. The second author is grateful to Bengt Nordström for providing necessary facilities to prepare a draft of this paper during his stay in Göteborg.

References

1. Ankrum, T.S., Kromholz, A.H.: Structured assurance cases: three common standards. In: Ninth IEEE International Symposium on High-Assurance Systems Engineering (HASE 2005), pp. 99–108 (2005)
2. Holloway, C.M.: Explicate78: uncovering the implicit assurance case in do-178c. Technical report 20150009473, NASA Langley Research Center (2015)
3. ISO/IEC: ISO/IEC Directives, Part 2, Principles and rules for the structure and drafting of ISO and IEC documents, 7th edn. (2016)
4. ISO/IEC/IEEE: 12207 FDIS Software life cycle processes (Final Draft International Standard registered for approval)
5. Knight, J.C., Rowanhill, J.: The indispensable role of rationale in safety standards. In: Skavhaug, A., Guiochet, J., Bitsch, F. (eds.) SAFECOMP 2016. LNCS, vol. 9922, pp. 39–50. Springer, Cham (2016). doi:[10.1007/978-3-319-45477-1_4](https://doi.org/10.1007/978-3-319-45477-1_4)
6. Moore, A.P., Klinker, J.E., Mihelcic, D.M.: How to construct formal arguments that persuade certifiers. In: Hinchey, M.G., Bowen, J.P. (eds.) Industrial-Strength Formal Methods in Practice. FACIT, pp. 285–314. Springer, London (1999). doi:[10.1007/978-1-4471-0523-7_13](https://doi.org/10.1007/978-1-4471-0523-7_13)
7. Tokoro, M. (ed.): Open Systems Dependability: Dependability Engineering for Ever-Changing Systems, 2nd edn. CRC Press, Boca Raton (2015)
8. Kinoshita, Y., Takeyama, M.: Assurance case as a proof in a theory towards formulation of rebuttals. In: Dale, C., Anderson, T. (eds.) Assuring the Safety of Systems, Proceedings of the Twenty-first Safety-Critical Systems Symposium, Bristol, UK, pp. 205–230 (2013). SCSC on Amazon ISBN 978-1-4810-18647

Computer Safety, Reliability, and Security
SAFECOMP 2017 Workshops, ASSURE, DECSoS, SASSUR,
TELERISE, and TIPS, Trento, Italy, September 12, 2017,
Proceedings
Tonetta, S.; Schoitsch, E.; Bitsch, F. (Eds.)
2017, XIV, 478 p. 138 illus., Softcover
ISBN: 978-3-319-66283-1