# ASsurance as Code (ASaC)

Shuji Kinoshita
*Advanced Institute of Industrial Technology*
Tokyo, Japan
kinoshita-shuji@aiit.ac.jp

***Abstract*—This position paper proposes a new concept of "Assurance as Code". Society has become increasingly complex, and many systems are interconnected (system of systems), so that a small bug can cause a major system failure. The need to describe assurance cases to improve system reliability is increasing, but the cost of describing assurance cases is very high. The solution to this problem is to develop an environment in which programmers can freely write assurance cases like program code. Just as the idea of "Infrastructure as Code (IaC)" enabled easy infrastructure construction with the help of gamification, this has the potential to make it easy for non-specialists to write assurance cases. The author is working on prototyping such an environment and outlines its current status and challenges.**

***Index Terms*—assurance case, IaC, ISO/IEC/IEEE 15026-2**

## I. Introduction

This position paper proposes a new concept of "Assurance as Code". Society has become increasingly complex, and many systems are interconnected (system of systems), so that a small bug can cause a major system failure. The need to describe assurance cases to improve system reliability is increasing, but the cost of describing assurance cases is very high.

The solution to this problem is to develop an environment in which programmers can freely write assurance cases like program code. Just as the idea of "Infrastructure as Code (IaC)" enabled easy infrastructure construction with the help of gamification, this has the potential to make it easy for non-specialists to write assurance cases. The author is working on prototyping such an environment and outlines its current status and challenges.

## II. Background

### A. Assurance 2.0 – Needs for the new assurance –

The paper [1] presents the need for a new assurance approach to address, for example, autonomous systems where key functions are driven by machine learning and AI. It allows for a more rigorous assurance case construction by framing inductive reasoning and argument rebuttal, while keeping the reasoning steps as deductive as possible.

### B. Standardization

A new international standard for assurance cases, ISO/IEC/IEEE 15026-2, was published in 2022[2]. This is a revision of the first edition published in 2011 and reflects the latest developments in assurance cases in recent years. Specifically, it standardizes the basis for mathematical definitions that allow for automatic consistency checking of assurance cases based on the idea of the formal assurance case [3], while organizing terms in notations such as GSN[4] and CAE[5] and in meta-models such as SACM[6].

### C. "XaC" paradigm

Although not necessarily so explicitly stated, the history of software development can be organized as "writing something in code" (X as Code). For example, the automation of testing, which used to be done manually by humans, is "Test as Code," and the coding of modeling, represented by PlantUML[7] and mermaid.js[8], can be called "Modeling as Code" or "Design as Code. The coding of infrastructure definitions using Docker[9] or terraform[10], which has become popular since the late 2010s, is called "Infrastructure as Code (IaC)" and has become widespread.

These "XaC" paradigms have not only increased the convenience of management through code, but have also demonstrated the possibility of making testing, design, and infrastructure construction, which were previously the work of specialists, the "work of programmers". The application of this to assurance is the attempt presented in this paper.

## III. Problem statement

Writing assurance cases involves significant human costs. There are two problems.

### A. Lack of human knowledge about assurance

The first is the lack of personnel who can write assurance cases in the first place. Assurance cases are still reserved for a few mission-critical systems, and the majority of system engineers (as far as the author knows, at least in Japan) do not even know they exist.

### B. Lack of automated tool support about assurance evolution

Second, there is a lack of tools to support the evolution of assurance cases. Assurance case creation tools include ASCE[11] and astah* System Safety[12]. Although these tools support diagrammatic drawing, they are not capable of automatically checking the consistency of vocabulary and arguments used in assurance cases. In addition, formal assurance case descriptions using the theorem prover Agda[3], [13] can check the consistency of vocabulary and arguments, but Agda is not always easy to learn, and the integrated development environment is not well-developed. The PVS[14] is also a similar situation. In addition, it is difficult to detect and automatically respond to changes in models and codes that

serve as evidence in either case. There is research on this by Matsuno et al.[15] but no practical tool yet exists.

## IV. Proposed solutions – New toolset for ASaC paradigm –

To solve these problems, the author plans to prototype and evaluate a new assurance case description environment. This is to realize a new concept of "Assurance as Code". This is an attempt to code things that have not traditionally been described in code, such as "Infrastructure as Code (IaC)". The description environment includes a new programming language for assurance case descriptions and its integrated development environment (IDE). These will be built using the Language Server Protocol (LSP) as a plug-in on Visual Studio Code[16]. The following functions are planned:

1) Programmatic assurance case description (by new programming language or DSL in host language)
2) Automatic generation of GSN and other representations from assurance case description language (SACM-based XML code generation)
3) Detection of changes by linking to evidence models and codes

All of these functions can be written as code, enabling change management using Git[17], etc., and assurance of the DevOps lifecycle, such as CI/CD. In addition, by enabling assurance cases to be written in an integrated development environment well known to programmers, the number of potential "writers" can be expected to increase.

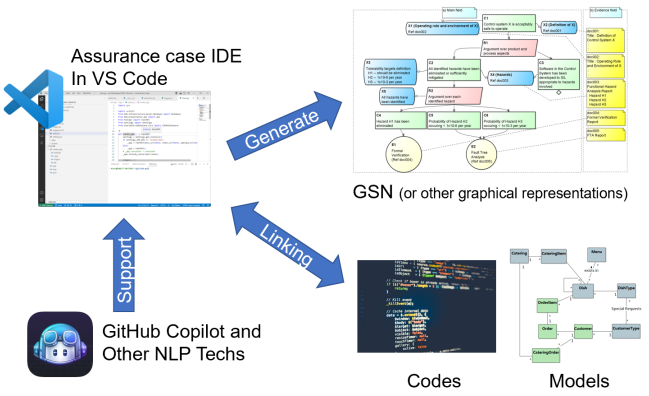


Fig. 1. Assurance as Code (ASaC) overview

In addition, by coding, it is also possible to support automatic code generation by AI. In other words, it will be possible to support assurance case writing through code creation support such as GitHub Copilot[18], and natural language processing can be applied to large-scale system development documents as input and a part of assurance case as output.

An overall view of these is shown in Fig. 1.

## V. Conclusion – Towards Assurance Development Game –

This paper presents a new concept called “Assurance as Code” (ASaC) and demonstrates its feasibility. Currently, the author is working on the architectural design of an assurance case description environment, with a comparative survey of existing results such as [1], [2].

Although there is no clear argument for this, many programmers must have found it “fun” to have various tests automated and to be able to build infrastructure with code. To begin with, programming is fun (even if sometimes painful). The various activities of system assurance will also become fun when they are coded, rather than simply being tedious tasks that require many man-hours.

The paper “The New New Product Development Game”[19] compared product development to a rugby “scrum,” which became the basis for later Agile development. The author believes that assurance development can also become like a kind of “game” of programming.

## References


[1] Bloomfield, R., Rushby, J.: Assurance 2.0: A Manifesto. aiXiv (2020). doi:10.48550/arXiv.2004.10474

[2] ISO/IEC JTC 1/SC 7.: ISO/IEC/IEEE 15026-2:2022 Systems and software engineering — Systems and software assurance — Part 2: Assurance case. ISO (2022). https://www.iso.org/standard/80625.html

[3] Kinoshita, Y. and 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, SCSC (2013)

[4] SCSC The Assurance Case Working Gruop.: Goal Structuring Notation Community Standard Version 3. SCSC (2021). https://scsc.uk/scsc-141c

[5] Claims Arguments Evidence.: CAE FRAMEWORK. Claims Arguments Evidence (2023). https://claimsargumentsevidence.org/

[6] Object Management Group.: Structured Assurance Case Metamodel Version 2.2. OMG (2021). https://www.omg.org/spec/SACM/2.2

[7] Roques, A.: Plant UML (2023). https://plantuml.com/

[8] Sveidqvist, K.: Mermaid – Diagramming and charting tool – (2023). https://mermaid.js.org/

[9] Docker Inc.. Docker. Docker Inc. (2023). https://www.docker.com/

[10] HashiCorp.: Terraform. HashiCorp (2023). https://www.terraform.io/

[11] Adelard.: ASCE Software Overview. NCC Group (2023). https://www.adelard.com/asce/

[12] ChangeVision, Inc..: astah* System Safety. ChangeVision, Inc. (2023). https://astah.net/products/astah-system-safety/

[13] The Agda Team.: The Agda Wiki. The Agda Team (2023). https://wiki.portal.chalmers.se/agda/pmwiki.php

[14] SRI.: PVS (Prototype Verification System). SRI International (2023). https://pvs.csl.sri.com/

[15] Matsuno, Y., Ishikawa, F., Tokumoto, S.: Tackling Uncertainty in Safety Assurance for Machine Learning: Continuous Argument Engineering with Attributed Tests. In: Romanovsky, A., Troubitsyna, E., Gashi, I., Schoitsch, E., Bitsch, F. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2019. Lecture Notes in Computer Science, vol 11699. Springer, Cham (2019). doi:10.1007/978-3-030-26250-1_33

[16] Microsoft.: Language Server Protocol. Microsoft (2022). https://microsoft.github.io/language-server-protocol/

[17] The Git community.: Git. The Git community (2023). https://git-scm.com/

[18] GitHub, Inc..: GitHub Copilot - Your AI pair programmer. GitHub, Inc. (2023). https://github.com/features/copilot

[19] Takeuchi, H., Nonaka, I.: The New New Product Development Game. Harvard Business Review (1986). https://hbr.org/1986/01/the-new-new-product-development-game