Assure-It: A Runtime Synchronization Tool of Assurance Cases

Shunsuke Shida, Atsushi Uchida, Masaki Ishii, Masahiro Ide, Kimio Kuramitsu

Assure-It: A Runtime Synchronization Tool of Assurance Cases

Shunsuke Shida, Atsushi Uchida, Masaki Ishii, Masahiro Ide, and Kimio Kuramitsu
Yokohama National University
Yokohama, Japan
{shida-shunsuke-vn, uchida-atsushi-vt, ishii-masaki-cs}@ynu.ac.jp , imasahiro9@gmail.com, kimio@ynu.ac.jp

***Abstract*—More recently, the idea of runtime synchronization of GSN has been proposed, where evidences are being collected from logs that are produced by monitors and other software components. By introducing the runtime synchronization, GSN can be regarded as a program where its validity is checked by applying runtime contexts. In this fast abstract, we introduce Assure-It, a novel tool that enforces administration scripts with assurance cases guidance and runtime synchronization.**

Keywords—administration scripts; assurance cases; system dependability; software engineering supports;

I. INTRODUCTION

Scripting languages such as Bourne shell and Perl have been broadly used to perform system administration tasks, including system maintenance, system diagnosis, and failure response [1]. As these tasks are strongly related to the realization of system dependability (reliability, availability, etc.) requirements, software engineering supports for administration scripts are practically significant. However, most of these today's scripts are written in an ad hoc manner, unfortunately resulting in several causes of system failures.

Assure-It is a novel open source tool and developed in the JST/DEOS project to provide the means of modularizing scripting solutions for system administration under dependability requirements. The key idea is the use of assurance cases in order to associate dependability goals with administration tasks, which will be composed in a final executable script. Due to the specified association, the partial failure of script execution can be detected as an error from the associated dependability goals. In addition, Assure-It allows us to argue an incremental analysis of failure-case, which enables richer failure/error handling.

This fast abstract will show how Assure-It works with the concept of assurance cases. Several dependability goals (such as system and data availability, privacy and accountability) are argued over assurance cases, by associating monitoring and administration tasks. From the assurance cases, Assure-It can generate an executable script, directly connected to the realization of argued dependability goals.

The rest of this fast abstract proceeds as follows. Section 2 introduces Goal Structuring Notation, a standard notation of assurance cases, used in Assure-It. Section 3 overviews how Assure-It generates executable scripts from the arguments on assurance cases. Section 4 briefly describes the summary of this fast abstract.

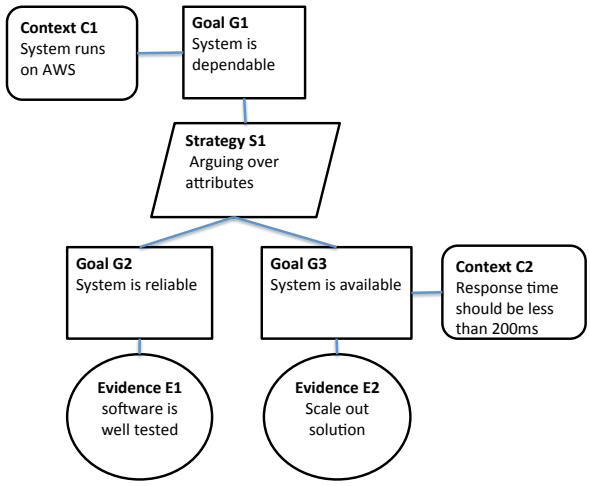

Fig. 1 An Example of GSN.

II. GSN AND THE CONCEPT OF ASSURE-IT

Assure-It has adopted GSN[2] as a common notation bridging existing assurance cases methodology. In addition, we attempt to add dynamic properties in order to synchronize assurance cases with runtime system through script executions.

A. Goal Structuring Notation

Goal Structuring Notation has four major elements, goal (depicted in rectangle), strategy (parallelogram), evidence (oval), and context (rounded rectangle). The goal element is a claim that a system certainly has some desirable properties. The evidence element is a fact supporting that the linked claim is true. The goal without linked evidence is called undeveloped goal and depicted with diamond. The strategy element is an assumption or a pre-condition that linked goal holds. Fig. 1 shows an example of GSN.

B. Extended Functional Evidence

The *functional GSN*, we propose in this fast abstract, is an extended one that accepts as one of valid evidence a program that produces logs supporting the correctness of its performance. Note that logging feedbacks are required for

runtime synchronization as depicted in Fig 2. A typical example of the program is a monitor notifying us of erroneous situations. Another typical example can be a system administration script that performs data backup and failure handling tasks, which straightforwardly lead to the realization of some dependability properties. From viewpoint of programming, these monitors and tasks are regarded as a function taking runtime contexts and then checking the validity of the associated goal.

It is important to note that the absence of failures in the functional element is not practical. Robin et al [3] proposes the meta assurance cases method to argue failure-case analysis on GSN. Using these methods, we can generate more reliable script that includes richer failure/error handling. Due to the space constrain, we are omitting the formalization of meta GSN in this fast abstract.

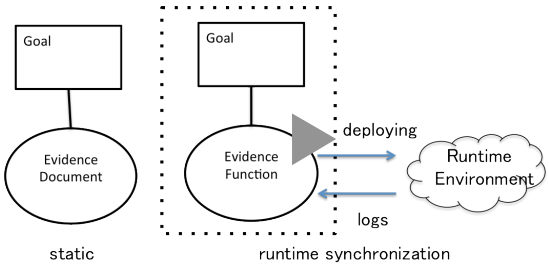

Fig. 2 Static Evidence and Runtime Synchronization

III. TOOL DESCRIPTION

Assure-It allows arguing the strategic division of dependability goals with assurance cases method. Fig. 3 describes the overview of Assure-It. It generates executable scripts from GSN arguments, binding each of operational solutions with strategy as control flows and deploys the generated scripts on running systems, and execution results are generated. Assure-It consists of two applications: a client and a server. Users create assurance cases by using client application, which is accessible through a mordern web browser.

A. Code Generation from Assurance Cases

As noted above, source code is generated from Assurance Cases, which is created by Assure-It to get in touch with runtime environment. In Assure-It evidence element consist of static evidence such as test results and code snippets, short length of shell scripts. The script that performs system administration is generated by making code snippets structured depending on rules described on strategy.

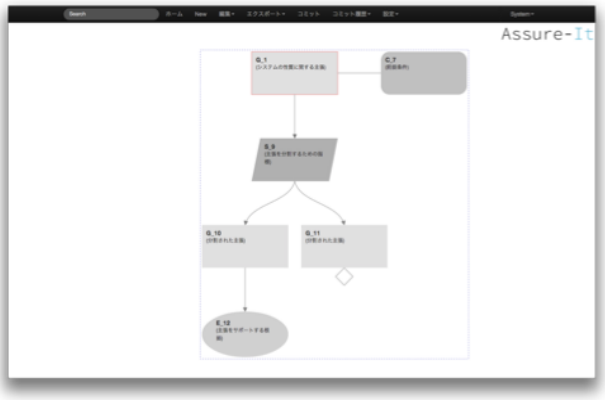

Fig. 3 An Overview of Assure-It

IV. SUMMARY

The assurance cases method provides the guidance to modularize administration tasks from viewpoint of dependability requirements. Assure-It is a tool that can generate an executable administration script by combining modularized tasks on assurance cases arguments. Running the generated script is an evaluation of dynamic assurance cases, being applied by runtime contexts.

REFERENCES

[1] Mario Tokoro. Open Systems Dependability: Dependability Engineering for Ever-Changing Systems. CRC Press, 2012.

[2] Tim, K. and Rob, W.: The Goal Structuring Notation A Safety Argument Notation, In Proc of DSN 2004 (2004).

[3] Robin E. Bloomfield, Peter Bishop: Safety and Assurance Cases: Past, Present and Possible Future - an Adelard Perspective, in Making Systems Safer: Proceedings of the Eighteenth Safety-Critical Systems Symposium, pp. 51-67, (2010).