Towards Compositional Assurance of Large Cyber-Physical Systems

Gabriel A. Moreno
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
gmoreno@sei.cmu.edu

Mark Klein
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
mk@sei.cmu.edu

Shambwaditya Saha
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
ssaha@sei.cmu.edu

Farzaneh Derakhshan
Illinois Institute of Technology
Chicago, Illinois, USA
fd@iit.edu

Limin Jia
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
liminjia@cmu.edu

Abstract

Assurance of evolving large cyber-physical systems (CPS) is time-consuming, and usually a bottleneck for deploying them with confidence. Several factors contribute to this problem, including the lack of effective reuse of assurance results, the difficulty to integrate multiple analyses for multiple subsystems, and the lack of explicit consideration of the different levels of trust that different analyses provide. In this paper, we present an approach to assure large CPS that aims to overcome these barriers.

ACM Reference Format:
Gabriel A. Moreno, Mark Klein, Shambwaditya Saha, Farzaneh Derakhshan, and Limin Jia. 2018. Towards Compositional Assurance of Large Cyber-Physical Systems. ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3722571.3727836

1 Introduction

Assurance of evolving large cyber-physical systems (CPS) is a bottleneck in deploying these systems with confidence and in a timely manner. Different factors contribute to this problem. One is the lack of effective reuse of assurance results. For example, a subsystem that has been previously assured should not need to be re-assured when integrated with other subsystems in a large CPS. Even within a single subsystem, the assurance results for some analysis domains should be reusable in subsequent versions when the analysis assumptions have not changed. Another factor is the inability to integrate multiple types of assurance analyses with different levels of trust. Analyses differ along several dimensions, including their domain (e.g., timing, safety), their method (e.g., human inspection, testing, theorem-based), the fidelity of the artifacts used as input (e.g., source code, timing model), and the trust or confidence we can have on their results. Another factor is the interdependence between the assurance of the different subsystems that comprise the CPS. For instance, the assumptions made by different subsystems could potentially contradict each other (e.g., the speed of a drone could have been assumed to be low for safety in one subsystem but high for efficiency in another).

Our research project in large-scale assurance (LSA) is developing an approach to overcome these barriers that slow down assurance by supporting the composition of assurance analyses and results in a representation we call *argument architecture*. With an underlying formal representation and logic system, an assurance architecture enables formal reasoning over such compositions. With this approach, it will be possible to automatically check that the assumptions made by different analyses are consistent and satisfied, and to assess the level of trust of an assurance result based on multiple analyses. This will enable focusing assurance efforts to achieve the desired level of trust in different parts of the system.

2 Compositional Assurance Approach

In the previous section, we described how analyses differ in many ways. LSA focuses on a higher level of analysis: the integration of results from diverse domain-specific analyses for different parts of the system; checking the logical soundness of the integration; and determining the trust level of composed analysis results.

The main building block of the argument architecture is a *judgment*, inspired by the notion of judgments in logic, which is a declaration that the expressed proposition is true. In LSA, a judgment expresses that a domain-specific analysis has shown a guarantee to hold at a given trust level under some assumptions. Visually, we represent a judgment as shown in Fig. 1. In the underlying logic, a judgment has the form $\Gamma \vdash A : [G]_{C@T}$, where A is a label for the analysis, Γ is the set of assumptions, G is the guarantee shown to hold for component C at trust level T.

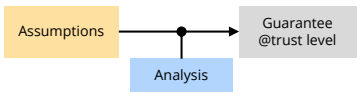

Figure 1: Visual representation of a judgment.

FACCT '25, May 6–9, 2025, Irvine, CA, USA

ACM ISBN 979-8-4007-1607-2/2025/05.
https://doi.org/10.1145/3722571.3727836

Currently, we use a trust concept based on *resources*, where a resource is anything that affects the result of an analysis for a given component, such as inputs, configuration parameters, or other properties. For example, for a stability analysis for a flight controller subsystem *FltCtl*, the resources could be the velocity and altitude of the drone. These resources define the dimensions of a space, and intuitively, the more of this space an analysis has covered, the more trust we can place on its results. Now, suppose that the analysis of the stability of *FltCtl* assumes that another subsystem provides reference points (i.e., fine-grained waypoints) at a frequency rp_f of at least 5Hz, we can write its judgment as $[rp_f \geq 5]_{C@T} \vdash A_{stability} : [stable(x)]_{FltCtl@\langle T, alt>200\rangle}$. In this judgment, C and T are placeholders for the component or subsystem C that will satisfy the assumption at a trust level T. This shows how an assumption's trust flows through the analysis and is combined with the trust level associated with the analysis ($alt > 200$). Next, we need a mechanism to compose judgments in the argument architecture and see how trust flows throw it.

From an analysis of the drone's software architecture, we know that the reference points are provided to *FltCtl* by the *Guidance* subsystem. Therefore, we know that a guarantee about the frequency at which this subsystem provides reference points is needed to satisfy the assumption of the previous judgment. In this particular case, a timing analysis is required to provide that guarantee. The timing analysis proves this under an assumption about the total CPU utilization on the computer where *Guidance* (*Gd*) executes. This assumption is that the sum of the utilization of *Guidance*, U_{Gd}, and all other tasks, U_o, does not exceed the bound $\bar{U}$. This can be expressed as $[U_{Gd} + U_o \leq \bar{U}]_{C_M@T_M} \vdash A_{timing} : [rp_f \geq 5]_{Gd@\langle T_M, T_1\rangle}$.

The underlying logic system has inference rules that explicitly handle trust levels. The soundness of the composition of the two judgments presented, shown in Fig. 2, can be formally verified by applying the *cut* rule. In addition to the assumption of the stability judgment being satisfied by the guarantee of the timing judgment, a sound composition also requires that the trust level of the resulting stability guarantee is not greater than the trust level of the guarantee that satisfies its assumption (i.e., $T \sqsubseteq T_1$). This shows how trust flows through the composition of judgments and consequently through the argument architecture.

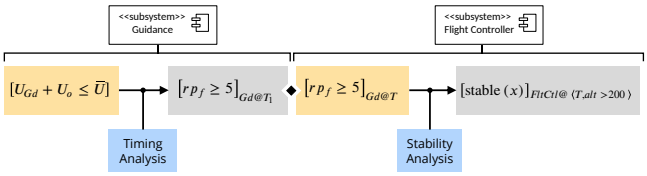

Figure 2: Composition of judgments.

With these elements and other inference rules in the logic system, the judgments are structured in the argument architecture to reflect the structure of the analyses that are used to reason about the assurance of a CPS. The argument architecture can then be checked to make sure that there are no unsatisfied or mismatched assumptions and that there are no logical inconsistencies. With the underlying representation of the judgments and their composition in the logic system, the argument architecture becomes a proof of the top-level claims or guarantees of the system that must be assured, and the formal analysis of the argument architecture is the checking of the soundness of that proof.

The explicit consideration of trust levels in the argument architecture, and their propagation through it allows not only seeing how different analysis contribute to or affect the overall trust in the guarantees, but also focuses assurance efforts were it matters.

3 Current and Future Work

The logic system to formally reason over the composition of assurance results has been developed. We are developing a tool to construct the argument architecture visually with the symbology used in this paper. It uses the OMG Structured Assurance Case Metamodel (SACM) for the underlying representation. There is still work to be done to address practical concerns for the adoption of the LSA approach, including a methodology to develop the argument architecture and capture the trust levels associated with its judgments. Finally, we want to extend the concept of trust to encompass other dimensions, including rigor and fidelity.

4 Related Work

Assurance cases are structured arguments to show how evidence supports claims that the system has some property, such as being safe or secure to operate [2, 3]. In the current state of the practice, the argument steps that decompose claims into subclaims are often inductive [1]. However, this does not ensure that the children formally imply the parent claims. To address this lack of soundness, researchers have proposed adding formality by making argument steps deductive, such that a claim is the logical consequence of its subclaims [1, 4]. Notwithstanding, there is also a recognition that having a completely formal assurance argument may be costly [3] and impractical [4]. LSA judgments hide the details of how an analysis was done, supporting less formal approaches like human inspection, but provide a composition interface so that the result of composition of the analyses can be formally checked.

5 Conclusion

We have provided a brief overview of LSA, an approach for assurance of large CPS. LSA aims to overcome some of the barriers that slow down assurance by hiding the details of the different analysis while allowing their sound composition using a logic system that supports rely-guarantee reasoning with trust levels. We believe that a methodology that integrates the co-design of the software architecture with its argument architecture can provide solid foundations for developing CPS that can be confidently assured.

Acknowledgments

Copyright 2024 Carnegie Mellon University and Farzaneh Derakhshan. This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center, and NSF under Grant No. 2350217. [DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution. DM25-0301

References

[1] Robin Bloomfield and John Rushby. 2020. Assurance 2.0: A Manifesto. doi:10.48550/ARXIV.2004.10474

[2] John Goodenough, Charles Weinstock, and Ari Klein. 2015. *Eliminative Argumentation: A Basis for Arguing Confidence in System Properties.* Technical Report CMU/SEI-2015-TR-005. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

[3] Patrick John Graydon. 2015. Formal Assurance Arguments: A Solution in Search of a Problem?. In *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. 517–528. doi:10.1109/DSN.2015.28

[4] Torin Viger, Rick Salay, Gehan Selim, and Marsha Chechik. 2020. Just Enough Formality in Assurance Argument Structures. In *Computer Safety, Reliability, and Security*, António Casimiro, Frank Ortmeier, Friedemann Bitsch, and Pedro Ferreira (Eds.). Springer International Publishing, Cham, 34–49.