Assurance-Directed Design of Cyber-Physical Systems*

Devesh Bhatt[1], Gabor Madl[1], David Oglesby[1], Sam Owre[2] Natarajan Shankar[2], and Ashish Tiwari[2]

[1] Honeywell Aerospace Advanced Technology, 1985 Douglas Dr. N, Golden Valley, MN 55422

[2] Computer Science Laboratory, SRI International, Menlo Park, CA 94025, USA

Transportation systems are moving from individual vehicles operating under manual control, either by a driver or a centralized traffic controller, to interconnected systems with varying levels of interaction and autonomy. In such systems, the complexity of the interaction can lead to emergent behaviors that are difficult to predict or anticipate. Before such systems can be deployed, we need a strong argument that it is possible to engineer them to the desired levels of safety and trustworthiness. Even with existing systems, we have seen a spate of errors arising from the interaction between different features and subsystems.

An assurance case presents an argument that an engineered artifact is fit for a purpose. In addition to the functional and performance characteristics of the artifact, an assurance case also argues that the artifact is reliable, safe, and secure. Traditionally, assurance cases have either been based on following a set of practices or have relied heavily on careful documentation. We have been developing an approach to assurance cases for transportation systems based on formal procedures integrating multiple tools for analysis and synthesis. We have also developed individual tools that operate at all levels of the design process. We argue, based on our experience, that

1. For cyber-physical systems like transportation systems, the assurance argument must be developed as part of the design. System safety should not be an isolated analysis but should be an integral part of the verification activities at the component and system levels. This would allow risks to be identified and resolved early in the design lifecycle.
2. The assurance case must come with both formal and empirical guarantees where a significant portion of the assurance processes are automated. This would allow assurance cases for large and complex systems to be curated and maintained even as the requirements and designs change and evolve.
3. Assurance is an enabling technology for developing systems of greater scale and complexity without compromising on safety and security. This kind of scaling can be achieved by decomposing the assurance argument horizontally

* This work was supported by NSF Grant CSR-EHCS(CPS)-0834810, NASA Cooperative Agreements NNA10DE73C and NNA13AC55C, and by DARPA under agreement number FA8750-12-C-0284. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF, NASA, DARPA or the U.S. Government.

by levels of abstraction, and vertically by function, to yield reusable patterns of design and assurance.

Assurance-Driven Design. Cyber-physical systems are composed of many interacting components, including both electro-mechanical components and software-based electronic control units. These components might operate at different physical and time scales, and can interact both physically through sensors and actuators, and electronically. Examples of such systems can range from simple thermostats to large complex systems like the power grid and the air-traffic control system. Software failures or vulnerabilities in cyber-physical systems can cause physical harm. Since these systems exhibit a possibly uncountable range of executions and operate in an unpredictable physical environment, the software must be accompanied by evidence of trustworthiness that goes beyond the empirical. The assurance case for such a system is *"a documented body of evidence that provides a convincing and valid argument that a specified set of critical claims about a system's properties are adequately justified for a given application in a given environment."*[3] This means that the software must not only work reliably, but it must be predicted to do no harm. Such an argument cannot be constructed after the software has already been built since the design of the software impacts the complexity of the assurance case.

Assurance-driven design takes a systematic approach to designing embedded software in a way that supports effective assurance. The design is decomposed into a series of abstraction layers that make certain explicit assumptions and approximations. At the highest level is the actual physical view of the system, e.g., a building climate control system. At the next level, we have the models that capture the engineering approximations, inaccuracies, and imprecisions. Such a model can be captured in a form that is amenable for simulation. The third abstraction layer introduces computational elements that involve dataflow and temporal dependencies between the physical and computation components. The fourth layer maps the computation to a physical platform with numeric errors as well as timing jitters. The physical layer also introduces failure models for sensors, actuators, and controllers. The design can also decomposed in the vertical dimension to isolate specific aspects such as information flow, timing, numeric error, logging, monitoring, error handling, sensor fusion, and fault tolerance. An assurance case structured along the above lines introduces a systematic separation of concerns that allows the global view of a reliable climate control system to be decomposed to systems that control heaters, air conditioners, and valves to regulate the temperature across the different sections of a building. Each level of the design employs models that abstract and approximate the inaccuracies, imprecision, jitter, and failures of the design elements at the lower levels. Some of these relationships across levels are informal or semi-formal, but a majority of them can be formalized. Once formalized, the claims associated with these relationships can be verified using one or more formal tools.

[3] R. E. Bloomfield, P. G. Bishop, C. C. M. Jones, and P. K. D. Froome, *Adelard Safety Case Development Manual*, 1998.

We are developing prototypes of such layered assurance arguments for a cockpit display subsystem, a simple air-traffic algorithm, and a high-assurance robot vehicle. As part of this work, we are also developing and sharpening the tools used for constructing assurance cases. Some of these tools deal directly with model-based designs and their high-level properties, while others deal with the source and object code. We have also developed a tool integration platform, the Evidential Tool Bus (ETB), specifically targeting the curation of claims and evidence from multiple analysis and synthesis tools. ETB can be used to integrate new tools, develop assurance workflows, apply these workflows to establish claims that are supported by arguments and evidence. Claims can be established directly by formal or semi-formal tools, by evidence in the form of documents, or by chains of inference. The certification objectives and verification claims and evidence required by directives such as RTCA DO-178C can be codified by means of rules in ETB. The argument and (version-controlled) evidence can be developed to meet these directives, and maintained against changes to the inputs, e.g., requirements, designs, or source code.

Assurance Tools and Techniques. There are now many formal and semi-formal techniques that are effective for achieving and maintaining high levels of assurance. These include abstract computational models that are suitable for cyber-physical systems, formal models for real-time and hybrid systems, model-based design tools, code generation tools, contract languages, test case generators, static and dynamic analyzers, and high-assurance platforms. Recent advances in Boolean and theory satisfiability, particularly through powerful solvers like Yices and Z3, have made it possible to generate test cases, synthesize controllers, build abstractions, and perform static analysis. Breakthroughs in solvers for nonlinear arithmetic constraints are particularly helpful in analyzing constraints from physical systems. Interactive theorem provers like Coq and PVS have been used to model and verify complex air-traffic algorithms and to verify hardware and compilers. We have also developed tools for directly analyzing Simulink models for dimensional correctness, signal ranges, and input/output contracts.

Challenges. Though computers have been used in embedded, real-time applications for decades, cyber-physical systems offer a bigger challenge since they are complex, distributed systems with many interacting elements. We are only at the preliminary stages of developing a discipline of assurance-driven design for such systems. We need a way of quantifying the impact of assurance-driven design on the cost of developing and assuring cyber-physical systems. We also need better automated tools, particularly those that can deal with controller synthesis and the analysis of large hybrid systems. A formal understanding of human factors can be very helpful in understanding the sources of human error and in designing interfaces that make it safer and easier to monitor and control large systems. Since cyber-physical systems, particularly those used in transportation, combine many different modules, it is important to develop compositional approaches to assurance-driven design that allow these techniques to scale with respect to system size and complexity. Developing an assurance case is an expensive exercise, and the cost needs to be amortized through reuse. However, as

illustrated by the Ariane-5 launch disaster, reusing designs and software can be fraught with risk. A rigorous discipline of assurance-driven design can go a long way toward mitigating the risk, promoting reuse, and reducing the cost of developing software-intensive solutions to complex transportation system challenges.