High-Confidence Cyber-Physical Co-Design

David Broman

University of California, Berkeley, USA and Linköping University, Sweden
broman@eecs.berkeley.edu

1. INTRODUCTION

Cyber-physical systems (CPS) [4] are characterized by combining computations, networks, and physical processes. Engineering cyber-physical systems is not new; high-end automobiles have for decades included complex embedded systems that interact with the physical environment. As an intellectual discipline, however, CPS design poses both new opportunities and challenges. The rapid development of a CPS with high-confidence of its functional correctness is a *co-design problem*—the design of the cyber part (embedded control systems and networks) and the physical part influence each other. For instance, when designing an industrial robot, the thickness of the robot arms changes the physical behavior of the system; thinner arms have less inertia and can move faster, but introduce more flexibility and spring behavior, making the control algorithm harder to design. As a consequence, to meet increasingly challenging system-level objectives, the cyber and physical parts need to be concurrently designed.

2. CO-DESIGN CHALLENGES

Today, model-based design is a well established approach for the concurrent design of different parts of cyber-physical systems. Complex systems can be virtually modeled using languages and tools such as Modelica [6], Ptolemy II [3], and Matlab/Simulink. To achieve *high-confidence* co-design, however, there are two major challenges.

Firstly, the modeling languages and tools must be *expressive* enough to capture the dynamic semantics of the modeled system. Expressiveness is easy to achieve if no consideration is taken to the possibility of *analyzing* the model; an ANSI C program can model the dynamics of a very complex system, but is formally hard to analyze. Simulation is one form of analysis, where properties are checked by testing, but full coverage is often not possible. Formal verification, such as model checking [2], can prove properties of the model, but requires a less expressive model of computation (MoC). The challenge is to provide both expressiveness and analyzability.

Secondly, a necessary condition for high-confidence of CPSs is high *model fidelity*, meaning that the model accurately imitates the real system. The problem is to automatically synthesize the model's cyber parts, such that the simulated model and the behavior of the real running system coincide. Synthesizing the functional behavior can be done today; the main challenge is to guarantee the preservation of the *timing behavior*.

3. OUR APPROACH

We propose an integrated language and compiler based approach to address these challenges. Our work is based on an extensible host language called *Modelyze* [1] (MODEl and anaLYZE), where various MoCs may be embedded as domain-specific languages (DSLs). The key aspect of our approach is that both the definition of the DSL and the CPS models are defined within the same language—Modelyze. We are currently evaluating this embedded DSL approach for encoding different MoCs as well as how formal verification can be encoded in the same framework. As a consequence, we address the combined expressiveness-analyzability problem by enabling *extensibility* within the language; if certain semantics cannot be described within the framework, a user may embed a new DSL with the desired properties.

The second part of our approach concerns high-confidence model synthesis. In another project within our research group, we are developing a Precision Timed (PRET) infrastructure, including an intermediate language and an ARM-based PRET machine [5] with thread interleaved pipeline and scratchpad memories for predictability. Our objective is, within Modelyze libraries, to define both functional and timing semantics as a translation from a Modelyze DSL to a PRET intermediate language. We intend to evaluate our approach in the mechatronics domain, where both the model of the physical plant and the control system are defined in Modelyze.

Project funding: Swedish Research Council #623-2011-955.

4. REFERENCES

[1] D. Broman and J. G. Siek. Modelyze: a gradually typed host language for embedding equation-based modeling languages. Technical Report UCB/EECS-2012-173, EECS Department, University of California, Berkeley, June 2012.

[2] E. M. Clarke. Model checking. In *Foundations of software technology and theoretical computer science*, pages 54–56. Springer, 1997.

[3] J. Eker, J. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Sachs, and Y. Xiong. Taming heterogeneity - the Ptolemy approach. *Proceedings of the IEEE*, 91(1):127–144, 2003.

[4] E. A. Lee. Cyber Physical Systems: Design Challenges. In *Proc. of the 11th Symposium on Object Oriented Real-Time Distributed Computing*, pages 363–369. IEEE, 2008.

[5] I. Liu, J. Reineke, D. Broman, M. Zimmer, and E. A. Lee. A PRET Microarchitecture Implementation with Repeatable Timing and Competitive Performance. In *Proc. of the 30th IEEE International Conference on Computer Design*. IEEE, 2012.

[6] Modelica Association. *Modelica - A Unified Object-Oriented Language for Physical Systems Modeling - Language Specification Version 3.3*, 2012.