Tutorial: Modeling, Verification, and Synthesis of Embedded Control Software

Samarjit Chakraborty[†] [*]
TU München
samarjit@tum.de

Pallab Dasgupta[†]
IIT Kharagpur
pallab@cse.iitkgp.ernet.in

Dip Goswami[†]
TU Eindhoven
d.goswami@tue.nl

S. Ramesh[†]
General Motors R&D
ramesh.s@gm.com

R. Venkatesh[†]
Tata consultancy services
r.venky@tcs.com

Majid Zamani[†]
TU München
zamani@tum.de

EXTENDED ABSTRACT

Traditionally, control applications are designed using a model-based approach – e.g., using Matlab/Simulink – where many idealistic assumptions about the implementation platform are made. These include the availability of infinite precision for mathematical computations, control laws being computed in negligible time, and delays from sensor to controller and controller to actuator being negligible. Models based on these assumptions are then used to automatically generate, e.g., C code representing the control applications. Such code is then partitioned into tasks, which are then mapped onto an, often distributed, architecture. In such architectures with multiple processors connected by a network of communication buses – e.g., as is the case in automotive architectures – the assumptions made at the model level are not true. Moreover, as implementation platforms become more complex, heterogeneous, and distributed, the gap between model-level assumptions and the implementation reality is continuing to increase. As a result, the control performance estimated at the model level deviates significantly from what is achieved in reality, which is referred to as the semantic gap between the model and the implementation.

In the past few years, validation and certification of control software has increasingly become important in various domains like automotive and industrial automation. While they are well established in domains like avionics, often they are at the expense of resource over-provisioning (e.g., because of higher sampling rates) that is not possible in more cost-sensitive domains like automotive. As a result, there has been a lot of recent interest on problems and solution techniques at the intersection of *control theory* and *embedded systems & software* – which constitutes one area of *cyber-physical systems*.

The first part of the tutorial will mainly discuss how to systematically translate control models into implementations. In particular, it will illustrate techniques for controller/architecture co-design [1, 2] and methods for rigorous verification and validation of control software [4], for example using model checking techniques [3, 5]. The distinguishing feature of this tutorial will be that it will address all layers of design abstraction, starting from models, to code and then finally the implementation of such code on concrete distributed architectures. Concrete examples from the automotive domain will be used to illustrate the presented techniques.

In the second part of the tutorial, we change the emphasis from validation and verification to formal synthesis. We show that it is possible to construct platform-aware correct-by-design embedded control software for the concrete systems without requiring any costly post facto validation and verification [7, 6]. One can leverage the proposed techniques to design controllers enforcing complex logic specifications on the original systems, difficult (or even impossible) to enforce using classical synthesis techniques. Examples of such specifications include properties expresses as formulae in linear temporal logic (LTL) or automata on infinite strings. Case studies from energy systems and motion planning will be employed to elucidate the results.

[*]Organizer, [†]Speakers

ESWEEK'14, October 12–17, 2014, New Delhi, India.

1. REFERENCES

[1] D. Goswami, T. Basten, and S. Chakraborty. Platform-aware design of embedded controllers. *ERCIM News*, 2014(97), 2014.

[2] M. Kauer, D. Soudbakhsh, D. Goswami, S. Chakraborty, and A. M. Annaswamy. Fault-tolerant control synthesis and verification of distributed embedded systems. In *IEEE Design, Automation & Test in Europe Conference (DATE)*, 2014.

[3] S. Mohalik, A. A. Gadkari, A. Yeolekar, K. Shashidhar, and S. Ramesh. Automatic test case generation from simulink/stateflow models using model checking. *Software Testing, Verification and Reliability*, 24(2):155–180, 2014.

[4] M. S. Prabhu, A. Hazra, and P. Dasgupta. Reliability guarantees in automata based scheduling for embedded control software. *IEEE Embedded Systems Letters*, 5(2):17–20, June 2013.

[5] A. Yeolekar, D. Unadkat, V. Agarwal, S. Kumar, and R. Venkatesh. Scaling model checking for test generation using dynamic inference. In *Proceedings of the 6th International Conference on Software Testing, Verification and Validation (ICST)*, pages 184–191, March 2013.

[6] M. Zamani, P. M. Esfahani, R. Majumdar, A. Abate, and J. Lygeros. Symbolic control of stochastic systems via approximately bisimilar finite abstractions. *IEEE Transactions on Automatic Control, forthcoming, arXiv: 1302.3868*, 2014.

[7] M. Zamani, G. Pola, M. Mazo Jr., and P. Tabuada. Symbolic models for nonlinear control systems without stability assumptions. *IEEE Transaction on Automatic Control*, 57(7):1804–1809, July 2012.