Demo Abstract: The ESMoL Modeling Language and Tools for Synthesizing and Simulating Real-Time Embedded Systems

Joseph Porter, Zsolt Lattmann, Graham Hemingway, Nagabhushan Mahadevan, Sandeep Neema, Harmon Nine, Nicholas Kottenstette, Péter Völgyesi, Gábor Karsai, and János Sztipanovits
Institute for Software Integrated Systems
Vanderbilt University
Nashville, TN 37235, USA
Email: jporter@isis.vanderbilt.edu
Telephone: (615) 343–4641

***Abstract*—High-confidence embedded real-time designs stretch the demands placed on design and development tools. We will demonstrate the design and testing of an embedded control system built using the ESMoL modeling language and supporting tools. ESMoL adds distributed deployment concepts to Simulink designs, and integrates scheduling analysis as well as platform-specific simulation. The testing system includes a simulated physical plant running in a hardware-in-the-loop configuration with the actual embedded controller.**

I. Introduction

Many classes of real-time embedded systems require assurance that engineering designs and implementations are safe, secure, and correct. Examples include software for medical devices, weapons control systems, and aircraft flight control. The time and effort (and consequently cost) of such assurance stands in contrast with the cost and schedule pressures of production engineering.

The ESMoL (Embedded Systems Modeling Language) domain-specific modeling language (DSML) provides user modeling constructs for designing embedded systems – including functional specifications in Simulink, models for distributed computing platforms, and deployment models assigning functions to tasks on computing nodes [1]. The language structure and the supporting Model-Integrated Computing (MIC) tools [2] enable the integration of analysis tools to provide design assurance for use in high-confidence designs.

The development workflow supported by the tools is depicted in Fig. 1. After the Requirements Specification has been created, designers start with familiar tools like Simulink to design and simulate the functional control system (Control Design). The Simulink design is imported into the modeling environment, and then annotated with Software Architecture and Component Design concepts. Platform models (Hardware and System Architecture Design) represent the hardware architecture. Componentized functions and signals are then assigned to tasks and messages deployed on the hardware (Software Deployment). From the composed model we can perform static real-time schedule determination and synthesize code for a distributed platform – assuming the platform provides clocked synchronous task execution and time-triggered messaging services.

The tools currently support development for unmanned aerial vehicles, in particular the Starmac quadrotor helicopter [3]. The control architecture is based on a fast inner-loop controller for attitude (here simplified to angle) and a slower outer-loop controller for position (here simplified to a scalar value). Our control technique is based on a composition of nested passive (PD) controllers to stabilize a process modeled as a cascade of continuous-time systems [4, Fig. 1].

II. Tool Features and Progress

The enabling ideas (beyond the existing capabilities of model-based development) are the integration of analysis and simulation tools to provide meaningful feedback to designers as early as possible in the development cycle, and a search for decoupling techniques which help make required analyses tractible. This demonstration will focus on the first aspect of our work (integrated analysis and simulation early in the

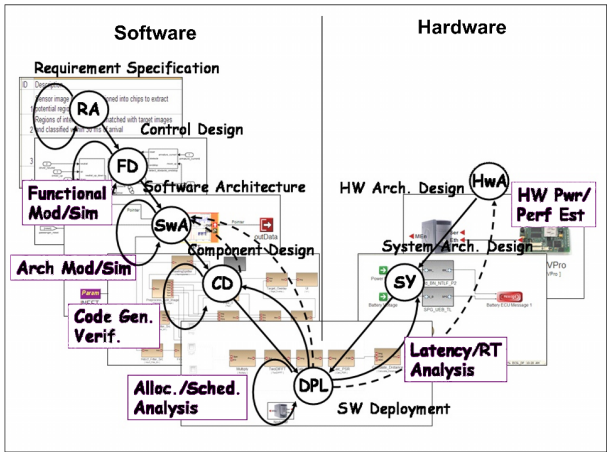

Fig. 1. Embedded development workflow supported by the tools.

design cycle) for embedded software synthesized from models. A detailed presentation of the design philosophy, intended uses and features, and limitations of the language and its tools is available in [1]. A few features are noteworthy:

1) The TrueTime toolkit [5] extends Simulink with blocks for modeling distributed platforms to simulate uncertainties due to distributed processing. Our modeling tools allow resimulation of the design using TrueTime blocks synthesized from the deployment model. A single design model targets both the continuous-time platform simulation environment of TrueTime and the tasks in the actual distributed system. Platform-specific simulation exposes potentially destabilizing effects of communication delays before physical system construction.
2) Platform communication delays are captured explicitly in the models. An integrated scheduling tool uses these timing parameters for analysis. Computed schedules are fed back into the modeling environment for TrueTime simulations and for the final physical deployment. Designers can iterate over possible component and plaform design alternatives and check schedulability.
3) In contrast to code generation tools like Real-Time workshop [6] – which generate code for individual processing nodes – ESMoL generates the task configuration code as well as the communication glue code for the distributed platforms on which it runs. The network may be a heterogeneous collection of processors and buses, as long as processing node and communication link types are supported by the code generators.
4) The entire tool infrastructure has been built around a single abstract semantic model, greatly simplifying the process of integrating additional analysis and synthesis tools. This ensures that all integrated tools have a single, consistent view of the modeling language semantics.

Current development includes extension of the TrueTime simulation generator, investigations into automated quantization of the digital controllers (subject to safety constraints), and consideration of other tools for verification of control properties and deadlocks. In particular, integration of new tools will require expansion of the language to include specification of richer operational models.

III. Tool Demonstration

The demonstration will display the following aspects the ESMoL modeling language and tools:

1) Synthesis of a Simulink control design from the GME modeling environment [7] to run on a simple distributed embedded processing system (Gumstix processor with an attached Robostix I/O processor [8]).
2) Computation of a cyclic real-time schedule using the integrated Gecode constraint solver [9] [10].
3) Execution of the synthesized control functions on a portable time-triggered virtual machine (FRODO [11]) running on the embedded hardware.
4) Simulation of the control loops using a hardware-in-the-loop (HIL) simulator communicating with the embedded system through a physical interface. The HIL simulator is a PC running the Mathworks' xPC target [6]. Fig. 2 shows the structure of the simulation system.

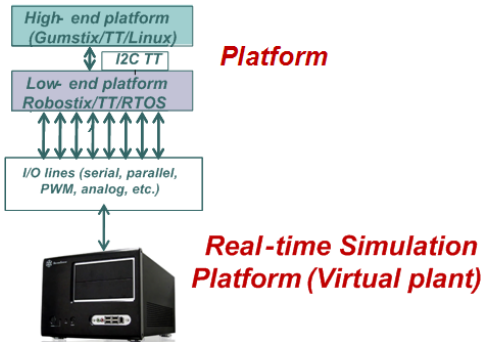

Fig. 2. Hardware-in-the-loop simulation architecture. The high-end PXA processor runs generic Linux, and the low-end AVR microcontroller runs FreeRTOS. The two communicate via an I^2C bus using a lightweight time-triggered messaging protocol. Communication to the plant simulator PC occurs over multiple high-speed serial lines.

Acknowledgment

This work is sponsored in part by the National Science Foundation (grant NSF-CCF-0820088) and by the Air Force Office of Scientific Research, USAF (grant/contract number FA9550-06-0312). 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 the Air Force Office of Scientific Research or the U.S. Government.

References

[1] J. Porter, G. Karsai, P. Volgyesi, H. Nine, P. Humke, G. Hemingway, R. Thibodeaux, and J. Sztipanovits, "Towards model-based integration of tools and techniques for embedded control system design, verification, and implementation," in *Workshops and Symposia at MoDELS 2008, Springer LNCS 5421*, Toulouse, France, To appear 2008.

[2] G. Karsai, A. Ledeczi, S. Neema, and J. Sztipanovits, "The model-integrated computing toolsuite: Metaprogrammable tools for embedded control system design," in *Proc. of the IEEE Joint Conference CCA, ISIC and CACSD*, Munich, Germany, 2006.

[3] G. Hoffmann, D. G. Rajnarayan, S. L. Waslander, D. Dostal, J. S. Jang, and C. J. Tomlin, "The stanford testbed of autonomous rotorcraft for multi-agent control," in *The Digital Avionics System Conference 2004*, Salt Lake City, UT, November 2004.

[4] N. Kottenstette and J. Porter, "Digital passive attitude and altitude control schemes for quadrotor aircraft," Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN, Report, 11/2008 2008.

[5] M. Ohlin, D. Henriksson, and A. Cervin, *TrueTime 1.5 Reference Manual*, Dept. of Automatic Control, Lund University, Sweden, January 2007, http://www.control.lth.se/truetime/.

[6] T. M. Inc., "Simulink/Stateflow Tools," http://www.mathworks.com.

[7] A. Ledeczi, M. Maroti, A. Bakay, G. Karsai, J. Garrett, C. T. IV, G. Nordstrom, J. Sprinkle, and P. Volgyesi, "The generic modeling environment," *Workshop on Intelligent Signal Processing*, May 2001.

[8] "Gumstix, inc. verdex and robostix i/o processing modules," http://www.gumstix.com.

[9] C. Schulte, M. Lagerkvist, and G. Tack, "Gecode: Generic Constraint Development Environment," http://www.gecode.org/.

[10] K. Schild and J. Würtz, "Scheduling of time-triggered real-time systems," *Constraints*, vol. 5, no. 4, pp. 335–357, Oct. 2000.

[11] R. Thibodeaux, "The specification and implementation of a model of computation," Master's thesis, Vanderbilt University, May 2008.