Simulation-Driven Architecture in the Engineering of Real-Time Embedded Systems

Trevor W. Pearce (pearce@sce.carleton.ca)

Abstract

The Simulation-Driven Engineering (SDE) research project aims to improve real-time embedded system development by integrating the use of modeling and simulation as a fundamental cornerstone in all development stages. Short-term goals are to simplify the reuse of engineering work products, and to develop a discrete-event based simulation architecture suitable for target products. The long-term goals include supporting ambitious simulation applications such as simulation- based acquisition. SDE is a new initiative, and this paper introduces the motivation for the research, the initial decisions that have shaped the current directions, and progress.

1. Introduction

Systems engineering methods approach product development in terms of a set of interacting components. This often involves modeling at multiple levels of abstraction, and consistency among work products is essential. Simulation in this context requires a component-oriented architecture that can be applied at various levels of abstraction. The architecture must allow simulations to interoperate as components of a larger system, allow simulation artifacts to be easily associated with target system components, encourage reuse of work products, and permit the inclusion of COTS components.

The Simulation-Driven Engineering (**SDE**) research project is a new collaborative initiative being carried out with Dr. Gabriel Wainer [GW]. SDE aims to integrate the use of modeling and simulation as a fundamental cornerstone in all aspects of real-time embedded system engineering. Although modeling and simulation is already a major component of many approaches, SDE goes further by proposing that a discrete-event simulation architecture also be used as the final target architecture for products. The research goals include incremental development, the reuse of work products, and the support of long-range applications.

2. Engineering Context

SDE is motivated by engineering concerns:

Systems: Systems engineering spans the "cradle to grave" lifecycle of a system. The breadth and timelines of system lifecycles require well-defined and integrated management processes that address system development, deployment, evolution, and retirement. The complexity of these lifecycles and processes has given rise to a need for research that strives to improve the way classes of problems and solutions can be engineered in a systems context.

Models: Models are abstract representations of concepts of interest. Models are abstract because they intentionally ignore some details in favour of emphasizing others. Modeling is a key aspect of all engineering methods. Engineers use models to focus communication, to analyze concepts, to record decisions, and to act as crucial work products that delineate progress in processes. Often, concepts must be modeled at various levels of abstraction that are appropriate to different stages in processes, and may be modeled using different views that are intended to draw out different subsets of related properties.

Simulation: A simulation is a dynamic activity in which a model is exercised for a specific purpose. A computer simulation involves the use of computers to execute programs that encode a model. Simulations are often used in engineering processes to validate concepts, verify models, and to reveal new insights into concepts through experimentation. In some applications, for example a pilot training simulator, a simulation may be the target product.

M & S: In this paper, the term "modeling and simulation" (M & S) is applied to the practice of constructing models in a way that simplifies the execution of the models as computer simulations.

Simulation Architecture: Discrete-event M & S is grounded in a perspective which views time as a partially ordered sequence of events. The events trigger changes to state variables that are

used to quantify both static and dynamic properties. Discrete-event models are employed in a broad spectrum of engineering domains, such as digital electronics, control systems, human/machine interfacing, training systems, software systems, and information systems. The architecture of a discrete-event simulation often encodes state variables as data, and encodes responses to events as programmed changes to state variable values. Some key factors in these simulations involve the management of time in ways that are relevant to the models, the generation and scheduling of events to be performed within the modeled time frames, and the handling of modeled input and output events as computer-related events.

Real-Time Systems: The engineering of real-time and embedded products presents significant challenges. Real-time performance requires timely response, often utilizing priority-driven control flow. The embedded nature of the systems often results in application-specific hardware components, and may entail the codesign of both hardware and software. The deployment and support of the systems must often account for unique aspects of the larger structure in which the system is embedded. Testing under actual operating conditions may be impractical, and in some cases impossible (e.g. a satellite control system). M & S is often used in the early development stages of these products. However, when development reaches a point where the target platform and target environment dominate the development focus, the early models and simulation artifacts are often abandoned.

Long-Range Applications: Significant efforts are underway to increase the use of M & S in a variety of ambitious applications [DoD]. For example, training exercises that require real-time embedded systems, such as geographically dispersed vehicle simulators, to be integrated as components in a distributed simulation. The exercises include hardware-in-the-loop and/or human-in-the-loop capabilities. The goals of the simulation-based acquisition process [Kon00] include the ability to simulate the end use of different products (as subsystems) before making a decision about which product to procure. These applications may yield significant gains; yet there are substantial development hurdles that must first be cleared.

These applications require consistent versions of both the product plus one or more simulations of the product. Developing and maintaining multiple consistent artifacts is likely to increase the cost and difficulty of the development process significantly. Furthermore, as versions evolve, consistent versions of all of the artifacts must also evolve.

SDE research addresses real-time embedded system engineering concerns in which product simulations, possibly at multiple levels of abstraction and with multiple views, must be developed and maintained in addition to the product. Highlights of the proposed approach include targeting products to a discrete-event simulation architecture, embracing the use of M & S as a fundamental strategic approach in all stages in development, and reusing work products when possible. The results of the research will help to integrate and complement, not complicate, common development strategies such as rapid prototyping, incremental development, product architectures, reuse, traceability, standards, COTS, and formal methods.

3. Research Decisions

The following initial decisions have helped to guide the expanding the role of M & S.

State Machines: A key decision is selecting a modeling paradigm that lends itself to the work products involved at the various stages in the development of real-time embedded systems. State machines have been chosen, since they are simple, widely understood, and have been shown to be applicable from abstract system requirements through to the detailed levels of hardware and software components.

DEVS: State machine models must be described in a way that allows easy simulation. The Discrete Event System Specification (DEVS) method [DEVS00] has been chosen based on its intuitive modeling of state machines and proven applicability. DEVS has a sound mathematical foundation, practical simulation semantics, supports the hierarchical composition of models, and includes time as an explicit attribute in models. DEVS has been used to model continuous-valued, discrete-time and hybrid systems. DEVS extensions have modeled cellular automata, stochastic systems, and fuzzy systems. A variety of simulation tools allow the execution of DEVS models, for example CD++ [CD++02], and an international effort is seeking interoperability standards [STD]. Parallel and real-time simulators have also been constructed for DEVS models [e.g. CD++03, RTD01].

Furthermore, DEVS is gaining a foothold in industrial practice [Gli03].

HLA: SDE research requires an underlying architecture that is flexible and component-oriented. The High Level Architecture (HLA) standard [HLA00], has been chosen as the enabling architecture across development stages. The HLA supports interoperability by allowing simulations to share data, events, and the management of a common notion of time. The HLA also specifies the API for the supporting Runtime Infrastructure (RTI). The goals of this research are not constrained by the HLA, and the use of the HLA is being hidden whenever possible. This avoids forcing developers to be knowledgeable of the HLA, and simplifies the transfer of the results to other environments.

4. Recent Work

Preliminary research has explored aspects of: the reuse of simulation work products across incremental development stages, the application of DEVS to real-time embedded systems, CD++ as a DEVS simulation engine, and the HLA.

An early effort created an HLA-compatible framework for an RTOS [Mao02]. An RTOS kernel was developed, and process deputies enabled each process to participate and collaborate as an individual component. The deputies presented a POSIX-compliant process API, and hid the HLA interactions with the kernel. The goal of the work was to allow processes to take advantage of the distributed virtual environment created by the HLA, regardless of the target environment.

Several efforts have focused on modeling at the hardware/software interface, and the system-level coupling of hardware components:

Computer system components and their interactions have been modeled, and then simulated using the HLA. Initial work modeled components by assuming that bus-level interactions were atomic [Sag02]. This allowed component functionality to be the central focus, without the distraction of bus protocols. CD++ notation was used, and simplifying properties of the models allowed a simplified simulation engine. An extension to this work developed a method for generating bus-level interaction modules automatically from a description of the bus protocol [Kha03]. The extension allows the reuse of the original models by wrapping the models with bus-level behaviour. The resulting simulations account for bus-level transactions that cannot be simulated using the original atomic transactions. An attractive feature of the extension is that the wrapper algorithms can be automated to eliminate the need for application engineers to have knowledge of the HLA.

Research has also explored the incremental introduction of hardware into the simulation loop and the reuse of work products [Li03]. An HLA-based architecture was used to simulate three iterations of a spectrum analyzer, which consisted of an input component, a DSP component and a display component. In the first iteration, each component was simulated using software modules. The DSP component included C source code for a Fast Fourier Transform (FFT). In the second iteration, a software harness interfaced a DSP hardware module to the HLA, and thereby allowed the new hardware module to directly replace the original software DSP component. The second iteration reused (as originally written) the input and display components, and the FFT C code. A third iteration used a software harness to integrate an audio hardware module, and thereby replace the original software input component. Testing with audio sampling frequencies up to 22 KHz revealed that the HLA was easily capable of supporting real-time behaviour in all iterations.

An ongoing effort is applying DEVS and CD++ to model a DSP-based Voice over IP telephone at three levels of abstraction [XPS]. At the highest level, the system is modeled as a single black box. An intermediate level models the system as three major components, based on the architecture of the existing telephone. The lowest level refines the intermediate components in terms of the programmer's models associated with the actual internal hardware devices. In addition to providing an interesting case study in the consistency of multiple-levels of work products, the work is exploring the reuse of work products across levels, and the use of abstract models to verify the correctness of lower-level models. The models will also provide a performance analysis test bed for CD++ simulations.

Work in progress is developing a generic HLA-compatible wrapper for CD++ simulations. A major concern has been establishing the cooperative control of time management between the CD++ simulation engine and the HLA RTI. This cooperation is essential to allow the CD++ simulation to participate with other components that may also regulate and constrain time advancement.

An important aspect of SDE research is the integration of results into practical tools. The ongoing CD++ Builder project [CD++B] is integrating existing CD++ tools into an Eclipse-based platform [Eclipse] to create an integrated and extensible development environment. A key practical component in the SDE approach is the HLA RTI. The ongoing Java RTI project [XPS] is porting the design of the Georgia Tech FDK [FDK] to Java, and extending the functionality to a more complete RTI feature set. Current work is integrating the Java RTI into CD++ Builder. This will simplify the development and runtime support of HLA-compatible CD++ simulations.

Research is just beginning on hardware support for simulation. The plan is to use system-on-chip technology to extend a processor's architecture to include support for time management and the scheduling of simulation transitions. DEVS models are targeted as the applications to be supported by the extensions.

Another effort just getting under way is the real-time analysis of simulations. This work is extending the analysis of known scheduling techniques (such as EDF) to DEVS models and their implementations.

5. Conclusions

SDE research is addressing the use of simulation-driven architecture in the engineering of real-time embedded systems. The proposed approach maintains a consistent focus on modeling and simulation across the entire development process, and eliminates the need to abandon modeling and simulation artifacts when focusing on the target platform. Furthermore, by basing the final product on a simulation architecture, development can be organized as a sequence of refinements that evolve abstract simulations at the level of requirements through to precise implementation "simulations". The refinement-oriented simulation process enjoys rapid prototyping, encourages reuse, and lends itself to the development and maintenance of multiple consistent artifacts.

The value of reusing simulation work products cannot be over-stressed. The advantages include: shortened development time (the components do not require implementation), increased reliability (components have already undergone a degree of verification and validation), and reduced development risk (fewer unknown components).

REFERENCES

[CD++02] "CD++: a toolkit to define discrete-event models". G. Wainer. In *Software, Practice and Experience*. Wiley. Vol. 32, No.3.. November 2002. pp. 1261-1306

[CD++03] A. Troccoli, G. Wainer, "Implementing Parallel Cell-DEVS". In *Proceedings of 36th IEEE/SCS Annual Simulation Symposium.* Orlando, FL. USA. 2003

[CD++B] www.sce.carleton.ca/esg/CDppBuilder/

[DEVS00] Bernard P. Zeigler, Herbert Praehofer, Tag Gon Kim, "*Theory of Modeling and Simulation*", 2nd Edition, Academic Press, 2000

[DoD] DoD 5000.59-P, "*Modeling and Simulation Master Plan*", October, 1995

[Eclipse] www.eclipse.org

[FDK] www.cc.gatech.edu/computing/pads/fdk.html

[Gli03] E. Glinsky, G. Wainer, "*Studying the Performance of DEVS Environments*", Technical Report SCE-03-011, Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada, 2003.

[GW] www.sce.carleton.ca/faculty/wainer.html

[HLA00] *"IEEE Standard for Modeling and Simulation High Level Architecture (HLA)",* IEEE Std 1516-2000

[Kha03] Hossam Khalil, "Integrating Timing Diagram Protocols with HLA Simulations", M.Sc. Thesis, Carleton University, 2003

[Kon00] K. Konwin, Dave Thomen, **"**Simulation Based Acquisition: An Overarching View", In *Proceedings of the Fall 2000 Simulation Interoperability Workshop*, Orlando, September, 2000

[Li03] Lidan Li, "DSP Hardware Surrogate for the HLA", M.A.Sc. Thesis, Carleton University, 2003

[Mao02] Rui Mao, "*HLA Compliant Real-Time System Operating Simulation*", M.Sc. Thesis, Carleton University, 2002

[RTD01] Seong Myun Cho , Tag Gon Kim, "Real time simulation framework for RT-DEVS models", *Transactions of the Society for Computer Simulation International*, v.18 n.4, p.203-215, December 2001

[Sag02] Amir Saghir, "*Computer System Modeling At The Hardware Platform Level***",** M.A.Sc. Thesis, Carleton University, 2002

[STD] www.sce.carleton.ca/faculty/wainer/standard/

[XPS] www.sce.carleton.ca/esg/XPERTS/index.htm