Advancing Model Driven Development Education via Collaborative Research

Aniruddha S. Gokhale
Dept. of EECS
Vanderbilt University
Nashville, TN 37235
a.gokhale (at) vanderbilt.edu

Jeff Gray
CIS Department
University of Alabama at Birmingham
Birmingham, AL 35294
gray (at) cis.uab.edu

Abstract

Rapid advances in hardware, networking and software technologies are fostering an unprecedented growth in a number of complex, distributed applications. Time-to-market pressures and keeping product costs affordable are the driving forces that seek newer ways to develop these complex applications rapidly and inexpensively. Significant additional challenges result from the need for longer shelf lives of these applications, which require shielding them from the constant evolution in the underlying technologies. Model driven development (MDD) is becoming the key technology to address these challenges. Therefore, it is necessary to develop tools and techniques to educate and train the current and next generation of engineers, scientists and developers in the information technology (IT) world. This paper describes the experience of the authors leveraging their synergistic research in MDD to develop and teach courses in academia comprising a mix of graduate and undergraduate students. The paper also provides a gist of lessons learned and our planned efforts to integrate MDD more effectively in the curriculum.

1. Introduction

Software development processes are increasingly becoming demanding. For example, there is a growing need for software development organizations to innovate rapidly, provide capabilities that meet their customer needs, and sustain their competitive advantage. Adding to these demands are increasing time-to-market pressures, limited software resources and the need to keep costs down, which often force organizations to innovate by leveraging existing artifacts and resources rather than handcrafting software products from scratch. *Product-line architectures* (PLAs) [1] and *middleware* [16] that hosts these PLAs are promising technologies for addressing these demands. However, to meet the runtime functional and systemic (i.e., quality of service) demands of PLAs and their product variants requires optimizations to the middleware, optimal configuration of the middleware, the right deployment of middleware artifacts, and possible redeployment and reconfiguration of the infrastructure in addition to the adaptations in the application logic.

Addressing these issues via *ad hoc* techniques, such as those based on handcrafting the solutions is tedious, error prone and costly, while also not providing the means to formally verify the correctness of these approaches. The two most significant factors leading to these problems are:

- **Level of abstraction:** The level at which developers try to address these challenges is usually at a low-level of abstraction, such as at the level of code using non-portable platform-specific artifacts. These techniques not only make the task of addressing the application concerns extremely cumbersome and error-prone, it also affects portability and long-term maintenance.
- **Crosscutting nature:** The different challenges faced by next-generation IT applications require effective design and fine tuning, which crosscut multiple layers of the infrastructure and the application logic. For example, adjustments made at one layer of the infrastructure leads to unforeseen consequences at some other layer of the infrastructure, or may adversely affect application logic.

Therefore, it is necessary for the next-generation of engineers and scientists to be made aware of these challenges as well as to educate them about the right set of technologies that can be used to address these challenges. The same argument holds in training the current IT personnel. Model-driven development (MDD), which comprises a suite of (overlapping) technologies, such as domain-specific modeling languages (DSMLs) [12], aspect-oriented software development (AOSD) [5] and Generative Programming [2], has emerged as the key technology to address many of the crosscutting challenges of next-generation IT systems.

This paper describes how the authors have used their synergistic research activities, which were funded in part by the DARPA PCES program and recently by an NSF award, to transition their research into a MDD-related curriculum. Section 2 presents a brief overview of the synergistic research activities of the authors; Section 3 describes how the two authors have transitioned their synergistic research activities into academia; and Section 4 provides concluding remarks along with lessons learned and future outlook.

2. Overview of Our MDD Research Activities

This section describes the MDD-specific synergistic research activities at the author's institutions. Our research focuses on developing model-driven development tools for addressing the deployment and configuration-related challenges of distributed IT systems. In this context, the Vanderbilt author and his team has developed the CoSMIC [11, 7] MDD tool suite. The University of Alabama at Birmingham (UAB) author and his team has developed the C-SAW aspect model weaving tool [17], which is used synergistically with CoSMIC to weave deployment and configuration-related crosscutting concerns at the modeling level, as well as for scaling many of the CoSMIC models. In the following we briefly describe the two tool suites.

2.1. CoSMIC MDD Tool Suite

The *Component Synthesis using Model Integrated Computing* (CoSMIC) [6] tool suite is an integrated collection of MDD tools that address the key lifecycle challenges of IT systems. Figure 1 illustrates CoSMIC's MDD tool chain.

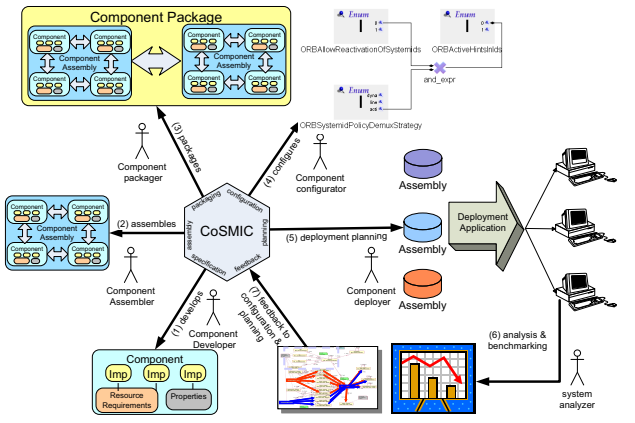

Figure 1. The CoSMIC MDD Toolsuite

CoSMIC supports modeling of IT system deployment and configuration capabilities, their quality of service (QoS) requirements, and adaptation policies used for IT system QoS management, particularly the real-time systems. The CoSMIC tools are implemented via *domain-specific modeling languages* (DSMLs) developed using the Generic Modeling Environment (GME) [14], which is a configurable toolkit for creating domain-specific modeling and program synthesis environments. CoSMIC uses GME to define the modeling paradigms (i.e., the syntax and semantics of a DSML [12]) for each stage of its tool chain. CoSMIC ensures that the rules of construction – and the models constructed according to these rules – can evolve together over time. Each CoSMIC tool synthesizes XML-based metadata that is used by the the CIAO QoS-enabled component middleware [18].

2.2. C-SAW Aspect Modeling Tool Suite

Our approach to aspect-oriented modeling (AOM) requires a domain-specific modeling weaver that processes the structured description of a visual model, which is different from traditional programming language weavers (e.g., the AspectJ weaver [13]) that support better modularization at a lower level of abstraction by processing source code.

We have designed C-SAW to provide support for modularizing crosscutting modeling concerns in the GME. This weaver operates on the abstract syntax tree of the model. To be effective, this weaver also requires the features of an enhanced constraint language. Standard OCL is strictly a declarative language for specifying assertions and properties of UML models. Our need to extend OCL is motivated by the fact that we require an imperative language for describing the actual model transformations. We designed a language called the Embedded Constraint Language (ECL) to describe model transformations. ECL is an extension of the OCL and provides many of the common features of OCL, such as arithmetic operators, logical operators, and numerous operators on collections (e.g., size, forAll, exists, select). A unique feature of ECL that is not provided within OCL, however, is a set of reflective operators for navigating the hierarchical structure of a model. These operators can be applied to first class model objects (e.g., a container model or primitive model element) to obtain reflective information needed in AOM.

The AOM approach that we have adopted in C-SAW can be summarized by the diagram in Figure 2. As shown in this figure, transformations are performed between the source models and the target models that belong to the same metamodel. C-SAW weaves additive changes into these source models to generate the target models relying on transformation specifications written in ECL.

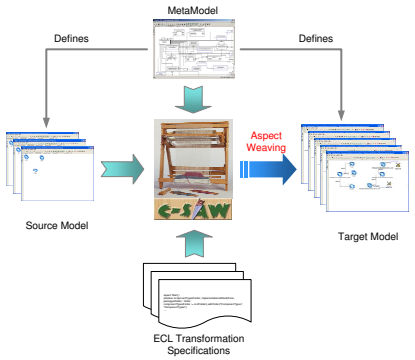

Figure 2. C-SAW Aspect Model Weaver Framework

3. MDD Classroom Education

This section describes how we have transitioned our research on MDD into the classroom.

3.1. Vanderbilt University Activities

This section describes how the Vanderbilt author is using MDD in his courses. Two of the courses were dedicated to the enhancement of MDD technology while the others have applied MDD technology.

3.1.1. Spring 2004 and 2005: MDD Courses The Vanderbilt author has in the past two years offered two special topics courses related to the theory and practice in model-driven development. Each of these courses were open to both graduate and undergraduate students. These courses were completely research-driven and used a format similar to the Paideia style of teaching (`www.paideia.org`). In this style of teaching, less emphasis is placed on the didactic aspects of teaching; instead more emphasis is on hands-on learning, seminar style, projects and discussions.

The Spring 2004 version of the MDD course was offered at a time when the CoSMIC tool suite was in its early stages of research and development. Our research team was interested in understanding the deployment and configuration challenges faced by IT systems that use different kinds of standard middleware solutions like J2EE, .NET and CORBA. The Paideia style of teaching was helpful in addressing these challenges. The didactic aspect, which is a small aspect of the Paideia style, dealt with introducing the students to the state of the art in MDD. However, the majority of the emphasis was on hands-on learning, group projects and class discussions. For example, the students were divided into project groups with some undergraduates partnering with graduates. Different groups focused on either different middleware solutions or different artifacts of deployment and configuration for a specific middleware.

The author's advisees who focused on investigating additional dimensions of their research through this course focused on sharpening the DSMLs and generators in CoSMIC. In particular, this included (a) enhancements to the PICML DSML used for component assembly and packaging, (b) the OCML DSML for modeling the configuration parameters for the middleware, and (c) the BGML DSML used to synthesize the empirical benchmarks to test the model-generated configurations and component assemblies. Other student groups focused on using the CoSMIC DSMLs for different case studies for different middleware. For example, a few student groups applied CoSMIC to an inventory tracking system (ITS) comprising software components for warehouse artifacts, such as forklifts, cranes, belts and operators.

Throughout the course, student groups would discuss their experiences with the rest of the class. The course comprised an end-of-semester student demonstrations that showcased the use of MDD for deployment and configuration of applications in the context of J2EE, .NET and CORBA Component Model. Moreover, at the end of the semester, the student's contributions on various aspects of the MDD investigations combined with their research activities resulted in four submissions to different conferences *e.g.*, ACM Middleware, IEEE RTAS, GPCE and OOPSLA Workshop publications. With the exception of the Middleware conference, all other submissions were accepted.

Building on the tremendous success of the Spring 2004 offering of the MDD course, the Spring 2005 course focused on investigating the use of MDD techniques for analysis and performance evaluation of IT systems. In this course, however, the undergraduates formed their own groups. We leveraged the Microsoft Windows Challenge worldwide competition as the means for the undergraduates to pursue their MDD course contributions. The graduate students focused on using model-driven analytical and simulation techniques for performance evaluation of IT systems.

Based on student feedback from the Spring 2004 semester, in the Spring 2005 offering, the author and the students used class time until the Spring break (about half the semester) for hands-on learning of the different MDD tools in class. This was very much in tune with the principles of Paideia and active learning. The tools studied comprised the Generic Modeling Environment (GME), Eclipse Modeling Framework (EMF) and a beta version of Microsoft Visual Studio 2005, which comprises the Corona DSL tool suite. In addition to the modeling tools, we covered various simulators, such as the NS-2 network simulator and Ptolemy environment, and performance analysis tools using Markov chains, such as

WinMVA. The purpose of this effort was to educate the students to use MDD tools to synthesize artifacts that can drive the backend analysis and simulation tools.

The undergraduate student groups used the E-boxes supplied by Microsoft to build custom images of the Windows CE environment, and applied it to different networked applications, such as crime "on-site" fingerprint match and lookup, ski area people tracking, and emergency response systems. Our experience with undergraduates was that they had to learn many new tools in one semester and hence their use of modeling tools was minimal. However, they attempted to use this collection of tools and provided a glimpse of how MDD could be utilized in different kinds of new emerging applications.

In the case of the graduate students, one group of students developed a DSML and generator for driving NS-2 network simulations. Another group developed a DSML and the associated generators to model Markov models of systems. Yet another group of students developed MDD tools for use in autonomic computing (in particular, for Enterprise Java Beans).

3.1.2. Fall 2004: Applying MDD in Operating Systems The Vanderbilt author teaches an undergraduate level operating systems (OS) course. In the Fall 2004 offering of the course, the author decided to integrate MDD principles into the course. The author's graduate student advisee was working on developing a simple model-driven OS simulator called VisualOS. At the time of the class offering, the VisualOS tool provided a model-driven front end that enabled students to configure simple memory management strategies, such as *first fit* and *best fit*. Virtual memory was not available at the time. Students were given a programming assignment that required them to set the desired memory management strategies. The generative tools within VisualOS synthesized the logic in C++ for the desired strategy. Students were then given a choice to modify the generated code. Our experience using VisualOS, student's feedback and its object-oriented design will be showcased as a poster and a demo [10] in the ACM OOPSLA Educator Symposium in Fall 2005.

3.1.3. Fall 2005: Applying MDD in Computer Networks With the successful offering of the special topics courses and the experimentation with using VisualOS in the OS class, the author plans to use more MDD-related tools and techniques in teaching computer networks in Fall 2005. He plans to use the OPNET IT Guru Academic Edition (`www.opnet.com`) to use a model-driven simulation engine to teach the fundamental ideas in computer networking. The OPNET IT Guru Edition provides a modeling front end comprising an embedded DSML that allows users to configure the type of the network and the desired topologies. A number of parameters are available to fine tune the topology, the properties of the synthesized discrete event simulator, and the result types and their visualization. The built-in DSML ensures a "correct-by-construction" network topology and the resulting simulator.

The Vanderbilt author plans to use an active learning approach by using half of the class time in didactic teaching of the networking concepts. In the remainder of the time, he will let students build simulations of the concepts taught in class and let them make different adjustments to the models, which will enable them to observe different traits in computer networking. The author is convinced that this approach will be very beneficial to understand systems and networking concepts that are the hallmark of current and next-generation distributed IT systems. It is expected that by the time of the symposium, the author will be able to share his experience using MDD techniques in the networking class. One additional outcome of this approach will enable the author to understand the crosscutting concerns that possibly pervade the realm of networking and network modeling. This will in turn help both the authors refine their R&D focus.

3.2. Univ. of Alabama at Birmingham Activities

At UAB, principles of MDD have been introduced to both undergraduate and graduate students. At the undergraduate level, Honors students have been mentored in a special course that provides a research experience throughout the academic year. At the graduate level, Masters and PhD students have the opportunity to take a course on reflective and adaptive software systems to explore concepts of meta-modeling and DSMLs. The following subsections describe these two efforts to introduce MDD techniques into the curriculum at UAB, as well as a description of a future goal of teaching basic modeling ideas to high school students.

3.2.1. Mentoring Honors Undergraduates in MDD Research Over the past two years, the UAB author has mentored four Honors undergraduate students in self-directed research projects that are focused on MDD. In each case, a pair of students were accepted into a summer research program sponsored by the National Science Foundation (in the USA, these are called NSF Research Experiences for Undergraduates - REUs). These REUs were held on other university campuses and the research performed during the summer was not related to modeling. When the students returned to the UAB campus for the academic year, they were asked to integrate MDD techniques with the summer projects.

1. Model-driven generation of Robotics Control Code: During Summer 2003, two female UAB students were accepted into a special summer internship at Vanderbilt to

learn about techniques to improve the development of embedded systems. Specifically, the students programmed Lego Mindstorms robots to accomplish various embedded control tasks. When the students returned to UAB, they were asked to apply advanced modeling techniques to model and synthesize embedded systems. The motivating problem for the research was the realization that hard-coded software for real-time embedded robotics control systems requires manual adaptation for each new configuration. In particular, the students developed domain-specific models that describe the configuration and layout of a hazardous environment, which is symbolically represented as an area contaminated with hazardous materials (*e.g.*, land mines), as well as objects to be rescued (*e.g.*, babies). The motivation was to model a disaster site that is too dangerous for humans to search for survivors. From the visual model specifications, model compilers were created to generate the embedded code to control two robots. The mission of the robots was to traverse the hostile terrain and locate the surviving babies.

2. Tailoring Mobile Devices from Models: During Summer 2004, a UAB student served as a summer intern on a project that investigated application tailoring of mobile devices. Specifically, the student project enabled a restaurateur to create an online menu for use on several different mobile devices. For the summer project, configuration for the various types of mobile devices was performed through XML configuration files. However, the XML files were tedious to modify and the lower-level of abstraction was a source of error because domain experts (*e.g.*, the restaurant owner) were not comfortable with XML. This project was a perfect fit for applying MDD to enable platform independent specification of the mobile application, with different model compilers generating the code needed to execute the application on different devices. The Honors research project created a meta-model for specifying application tailorability, as well as necessary model compilers. This provided a domain-specific modeling environment that allowed the restaurant owner to specify the essential details that they wanted to capture in their mobile menu, but in a way that removed them from the accidental complexities of mobile tailoring.

This mentoring relation resulted in a grant from the Computing Research Association for research supporting women (e.g., the CRA Research Experience for Women CREW). Two regional conference papers were produced by these undergraduates at the *ACM Southeast Conference*, ACM's longest running conference (please see [4, 3]). Additionally, two different students were awarded a top prize for their presentation of this work at the *ACM Mid-Southeast Conference*.

3.2.2. Integrating MIC into a Graduate Course on Reflection and Metaprogramming Over the past three years, the UAB author has taught a graduate level course on adaptive and reflective software. A focus of the course is on techniques that support adaptation of software either at compile or design time (static adaptation), and run-time (dynamic adaptation). The course covers implementation issues such as meta-programming, reflection, and AOSD.

After introducing reflection and metaprogramming, the students are taught modeling issues concerning metamodeling and development of associated model compilers. As a project for the course, students are provided with a description of a domain and asked to create a DSML for the domain, as well as the associated model compilers. Example projects in the past include: a domain for modeling a finite-state machine and a Java code generator; a domain for modeling extensions to Petri Nets and an associated C++ code generator; an entity-relationship modeling tool, along with a model compiler to generate database definition language statements.

After the students have constructed a DSML and associated instances, the concepts of aspect modeling are presented along with several examples using C-SAW. As a result of this course, five doctoral students have chosen MDD-themed dissertation topics. A Masters thesis has already been completed as an extension of the modeling ideas covered in the course.

3.2.3. Future Educational Opportunity: High School Students and MDD During Summer 2004, the UAB author sponsored a 7-week summer robotics camp where high school students programmed Lego Mindstorms Robots using Java (details are available at `http://www.cis.uab.edu/heritage`). This camp was expanded in Summer 2005 to support six students from the top magnet school in Alabama (this school was recently ranked by *Newsweek* magazine as the fourth best public high school in the USA). A goal of this outreach is to mentor the high school students throughout the year to prepare them for science fair projects. Through continued mentorship, the students will be introduced to modeling concepts and asked to adopt a model-driven approach to their robot project. This will be similar to the undergraduate Honors experience, but at a level appropriate for high school students. An experience report will be prepared later to document the ability of these students to grasp modeling tasks.

4. Conclusions

This paper describes our synergistic research activities and how we have transitioned them into courses at Vanderbilt University and the University of Alabama at Birmingham. Moreover, our experience using the MDD approach

and the tools we developed has helped to explore new dimensions of research. In the remainder of this section, we outline some of the lessons learned over the last two years teaching MDD and related concepts while conducting research in this area.

4.1. Lessons Learned

After offering the MDD courses and through his R&D on the CoSMIC MDD tools, the Vanderbilt author came to the conclusion that students were very happy with the style of the course offering. However, one thing that became evident was the wide range of tools available and the learning curve involved in mastering these tools within one semester. The result was that it became difficult for some groups (*e.g.*, those who did not have prior expertise on the MDD concept) to accomplish major goals.

We have found that students may experience an initial learning curve with respect to grasping the concepts of metamodeling. For those students who have enrolled in a database course prior to the modeling courses, it is easier to introduce the idea by relating to the correspondence between a database schema definition (which corresponds to a metamodel) and the specific extension of the database representing the values in database tables at a specific moment in time (which corresponds to instances of a metamodel). For those students who have not been exposed to the distinction between a schema definition and its instances, we have found it helpful to introduce such students to very simple metamodels, along with example instances and associated model interpreters.

For example, the first DSML that we introduce in a course is a very simple finite state machine (FSM). A simple FSM metamodel contains less than 5 modeling elements. From the FSM model, we ask the students to construct an FSM instance representing an automated teller machine. The FSM example can progress toward an introduction of model interpreters. During the first introduction, we typically provide a complete model interpreter for the FSM example. From this simple example, the students can then be taught how to use OCL constraints on a metamodel (in the GME, constraints are evaluated during model construction). After the students have mastered this simple DSML, we show them more complex examples representing a Petri Net and a network configuration modeling language.

¿From the UAB author's experience, the benefits of MDD can be better understood by students if they have been asked first to accomplish similar tasks in a manual manner. By understanding the accidental complexities that emerge in configuring and maintaining applications, the benefits of MDD can be appreciated. This was observed in the Honors research projects at UAB: students were asked to hard-code robotics control code and mobile device tailoring and then make changes to the platform configuration; the advantage of MDD was more evident when the same students had a modeling tool to make the same changes rapidly using a DSML.

The UAB author has also found great benefit from those graduate students who have enrolled in the modeling classes, but have different research areas. Many of the accidental complexities that are eased by a modeling approach can also be applied to research problems beyond the traditional software engineering context. As an example, two of the students who attended the adaptive systems class were from the high performance and scientific computing lab at UAB. They have applied the modeling techniques from the course into their own dissertation topics, resulting in two journal papers [9, 8]. The lesson learned from this observance is to realize the benefits in cultivating the collaboration among the students in other research labs.

At Vanderbilt University, there already is a graduate level course on Model Integrated Computing (MIC), which is a form of MDD. Students learn the concept of DSMLs and model interpreters in the context of the GME environment. Our recommendation is to teach some of these capabilities at the undergraduate level. As MDD becomes more ingrained in the development of IT systems, the authors envision that the concepts of DSMLs and MDD in general will have to be taught early on in the same spirit as the students are introduced to programming languages, such as Java.

In the Vanderbilt author's research, his team faced a number of challenges dealing with the crosscutting nature of changes that need to be made at the modeling level and also numerous challenges related to model scalability. The synergies between CoSMIC and C-SAW have alleviated these problems to a large extent.

4.2. Future Outlook

We are working towards broader dissemination of our experiences and the MDD technologies we have developed via forums, such as conference tutorials and demos. In conjunction with another collaborator (Dr. Swapna Gokhale of the University of Connecticut), we are exploring the use of MDD technologies for design-time performance analysis of composable middleware. We have recently been awarded a one year NSF grant to pursue this research agenda and are considering the possibility of summer exchange of students. As an initial investigation into this new area, we have developed a DSML that allows a user to model a system using stochastic reward nets (SRNs) [15] to conduct performance analysis of computer systems.

References

[1] P. Clements and L. Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley, Boston, 2002.

[2] K. Czarnecki and U. Eisenecker. *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley, Boston, 2000.

[3] V. Davis, J. Gray, and J. Jones. Generative approaches for application tailoring of mobile devices. In *Proceedings of the 43rd ACM Southeast Conference*, volume 2, pages 237–241, Kennesaw, GA, March 2005.

[4] R. Dennison, B. Shah, and J. Gray. A model-driven approach for generating embedded robot navigation control software. In *Proceedings of the 42nd ACM Southeast Conference*, pages 332–335, Huntsville, AL, April 2004.

[5] R. Filman, T. Elrad, M. Aksit, and S. Clarke. *Aspect-Oriented Software Development*. Addison-Wesley, Reading, Massachusetts, 2004.

[6] A. Gokhale, K. Balasubramanian, J. Balasubramanian, A. S. Krishna, G. T. Edwards, G. Deng, E. Turkay, J. Parsons, and D. C. Schmidt. Model Driven Middleware: A New Paradigm for Deploying and Provisioning Distributed Real-time and Embedded Applications. *The Journal of Science of Computer Programming: Special Issue on Model Driven Architecture*, 2005.

[7] A. Gokhale, D. C. Schmidt, B. Natarajan, J. Gray, and N. Wang. Model Driven Middleware. In Q. Mahmoud, editor, *Middleware for Communications*, pages 163–187. Wiley and Sons, New York, 2004.

[8] Z. Guan, F. Hernandez, P. Bangalore, J. Gray, A. Skjellum, V. Velusamy, and Y. Liu. Grid-flow: A grid-enabled scientific workflow system with a petri net-based interface. *Grid Workflow Special Issue of Concurrency and Computation: Practice and Experience*, 2005.

[9] F. Hernandez, P. Bangalore, J. Gray, Z. Guan, and K. Reilly. Gauge: Grid automation and generative environment. *Grid Workflow Special Issue of Concurrency and Computation: Practice and Experience*, 2005.

[10] J. H. Hill and A. S. Gokhale. Visual os: An object-oriented approach to teaching operating system concepts. In *ACM OOPSLA Educator Symposium Poster and Demo Presentation*, San Diego, CA, Oct 2005.

[11] Institute for Software Integrated Systems. Component Synthesis using Model Integrated Computing (CoSMIC). www.dre.vanderbilt.edu/cosmic, Vanderbilt University, Nashville, TN.

[12] G. Karsai, J. Sztipanovits, A. Ledeczi, and T. Bapty. Model-Integrated Development of Embedded Software. *Proceedings of the IEEE*, 91(1):145–164, Jan. 2003.

[13] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. V. Lopes, J.-M. Loingtier, and J. Irwin. Aspect-Oriented Programming. In *Proceedings of the 11th European Conference on Object-Oriented Programming*, pages 220–242, June 1997.

[14] A. Ledeczi, A. Bakay, M. Maroti, P. Volgysei, G. Nordstrom, J. Sprinkle, and G. Karsai. Composing Domain-Specific Design Environments. *IEEE Computer*, pages 44–51, November 2001.

[15] J. Muppala, G. Ciardo, and K. S. Trivedi. Stochastic reward nets for reliability prediction. *Communications in Reliability, Maintainability and Serviceability: An International Journal Published by SAE Internationa*, 1(2):9–20, July 1994.

[16] R. E. Schantz and D. C. Schmidt. Middleware for Distributed Systems: Evolving the Common Structure for Network-centric Applications. In J. Marciniak and G. Telecki, editors, *Encyclopedia of Software Engineering*. Wiley & Sons, New York, 2001.

[17] Software Composition and Modeling (Softcom) Laboratory. Constraint-Specification Aspect Weaver (C-SAW). www.cis.uab.edu/gray/Research/C-SAW, University of Alabama at Birmingham, Birmingham, AL.

[18] N. Wang, D. C. Schmidt, A. Gokhale, C. Rodrigues, B. Natarajan, J. P. Loyall, R. E. Schantz, and C. D. Gill. QoS-enabled Middleware. In Q. Mahmoud, editor, *Middleware for Communications*, pages 131–162. Wiley and Sons, New York, 2004.