USING ECLIPSE-BASED SOFTWARE TOOLS TO TEACH MODEL-BASED DESIGN METHODOLOGY IN A PROGRAMMING TOOLS COURSE AND AN EMBEDDED COMPUTER SYSTEMS COURSE

Nannan He, Han-Way Huang
Electrical and Computer Engineering Technology Department
Minnesota State University at Mankato, Mankato, USA

Abstract

This paper presents our experience of introducing both eclipse-based tools and model-based design (MBD) methodology into a system-level Programming Tools course for senior electrical engineering and computer engineering students. Eclipse is an integrated software development environment from IBM. Eclipse-based development tools have been widely employed by software projects in both academy and industry. Many eclipse-based software tools support MBD, an emerging development methodology for creating complex embedded software. We introduce students to the MBD process in combination with eclipse-based tools. The goal is to equip engineering students with the knowledge of advanced system development method and software tools so that they are able to utilize the tools for the efficient and cost-effective development of embedded systems. Our preliminary observations show that this combination could help students understand advanced software development technologies in practice, and improve the efficiency of designing and implementing embedded software projects.

Introduction

Nowadays, in many computing systems, the software portion is expected to have the greater impact on the behavior of entire systems. Knowledge of computing and software programming is important to engineering and technology students. The US Bureau of Labor Statistics predicts that computing will be one of the fastest-growing U.S. job markets in STEM through 2020: about 73% of all new STEM jobs will be computing related [1]. Moreover, software development training could be a valuable experience for students, as it can cultivate students' problem solving and process development capability.

However, programming is often considered to be difficult for engineering and technology students. They usually study the syntax and semantics of low-level programming languages such as C or assembly in one or two semesters. Compared with computer science major or software engineering major students, they have fewer opportunities to apply programming skills. It is common among engineering students that the language syntax is easily forgotten. When a class project involves software programming, students often spend a large amount of time in debugging syntax and semantics errors, with little time left for the algorithm development and verification. Many engineering students consider writing a small program with 300~500 lines of C code as a painful experience. A large percentage of junior or senior design projects that cannot be accomplished on time are due to the prolonged software implementation and testing stage.

Model-based design (MBD) is an emerging methodology for developing complex software systems, especially embedded software. Its efficiency has been demonstrated in software engineering. For example, Matlab/Simulink from MathWorks is a graphical programming tool for multi-domain simulation and MBD has become the predominant software modeling language in many motion controls, aerospace and automotive applications. By promoting the use of domain-specific notations to graphically represent specifications and designs, MBD can identify design flaws at the early stage and avoid costly design fixes during the late stage. The implementation of software systems can be

generated automatically or manually from high-level models. In recent years, multiple EU-funded projects on embedded systems have been launched or completed to promote the wide application of MBD in industry, and to solve challenges encountered in different real-world application domains, such as the MOGENTS project [2] (Model-based Generation of Tests for Dependable Embedded Systems), and the SESAME project (A Model-driven Test Selection Process for Safety-critical Embedded Systems) [3]. However, few universities in America currently offer engineering and technology students the knowledge of MBD.

In the last decade, eclipse-based tools have been widely applied to developing dependable embedded software systems in various applications such as automobile and automation industries. Eclipse is an integrated development environment (IDE) originated by IBM, which comprises a base workspace and an extensible plug-in system for customizing the environment. Eclipse has been widely used for teaching software programming in computer science courses as it has several important benefits for computer programming. Firstly, by means of various plugins, Eclipse provides multi-language programming, such as Java, C/C++. Secondly, Eclipse is cross-platform, which supports Windows, Linux and Mac OS as well. Thus, it is preferable for developing Linux-based embedded systems. Thirdly, Eclipse is free and open source under the terms of the Eclipse Public License. Most Eclipse-based software tools are also free for education purposes. In recent years, eclipse-based tools have been employed in increasing numbers of software projects in both academics and industry. They have become available for dealing with a wide range of embedded systems development problems, such as microcontroller (MCU) programming, system modeling, real-time computing platforms, and FPGA based embedded systems development. Moreover, the Eclipse modeling framework which is an eclipse-based modeling platform and code generation facility, is one of the most popular and well-known MBD initiatives.

Acronyms

CCS: Code Composer Studio
CDT: C/C++ Development Tool
EMF: Eclipse Modeling Framework
FPGA: Field Programmable Gate Array
IDE: Integrated Development Environment
MBD: Model Based Design
MCU: Microcontroller
PLC: Programmable Logic Controller
PT: Programming Tools
SoC: System on a Chip
TI: Texas Instruments
V&V: Validation and Verification

Programming Tools Course Description

The Programming Tools (PT) course is usually required for computer engineering or computer science majors, and other engineering and technology majors in many universities in America. It can be offered at either the introductory level or the system level. At the introductory level, the PT course emphasizes the basic methodology and tools supporting program compiling, linking, testing and debugging [4]. At the systems level, it typically focuses on the concepts of system-level programming (e.g., C/C++, Python and LabVIEW input language); tool chains for the group software development; and topics on software system design, implementation, testing strategies and documentation [5].

The PT course presented in this paper is closer to the systems level. It is organized as 2 hours of lecture and 2 hours of laboratory per week. At the end of the course, students are expected to be capable of utilizing programming tools to develop a complete hardware/software embedded system as their course project. In many cases, after initiating the project, students quickly move to the implementation stage after a brief design phase, and start the C programming and debugging iterations using an IDE. Although this approach works for the small-scale course project, students have reported that it is very time consuming and inefficient. And the behavior of the constructed

system often deviates from the original design plan. Educators have recognized the need of introducing efficient and cost-effective programming tools to students. The main goal is to equip students with the knowledge of developing complex engineering systems under a large number of constraints.

Experts in the software engineering and computer science communities advocate introducing the MBD methodology to students. Students are provided with insights, techniques and tools to alleviate the difficulties of developing complex software systems. Educators have either integrated MBD into an existing software design course [6], or proposed a new project-based course to solely teach MBD [7]. However, as these courses are mainly for computer science students, their contents are too theoretical for engineering and technology students who have a limited software development background.

Course Learning Objectives

The intent of our PT course is to convey the practical knowledge about programming to students. We added new materials on MBD from the engineering practitioner's point of view to this course with three top objectives which are further divided into sub-objectives.

1. To improve students' awareness of the latest MBD methodology.
   1.1 To identify basic stages of the MBD process.
   1.2 To understand the unique features of MBD (executable design specification, automated code generation, and continuous verification and validation).

2. To develop students' appreciation for MBD that contributes to the efficient and cost-effective system development.
   2.1 To comprehend the advantages of MBD compared with existing software development methods such as the V-model.

3. To introduce students to modern eclipse-based software development tools that support MBD.
   3.1 To utilize software tools for efficiently developing embedded systems.

Model-based Design Concepts

We introduce fundamental MBD concepts (that are important for an engineering practitioner) to our students during the first week. Five basic steps of MBD are covered, from the requirement analysis, system design, implementation, integration to continuous validation and verification (V&V). Based on the MBD process illustrated in Figure 1, we discuss the main differences between MBD and conventional software development processes like the waterfall model or V-model to encourage active learning. For instance, students can summarize by themselves that verification is conducted continuously during each of the other four steps, instead of at their completion. Students are also guided to learn new concepts along each basic MBD step. Using the *system design* step as an example, students study the concept of *Executable Specification* (in terms of models shown in Figure 1). It can unambiguously model the entire system functionality, including an environment, physical component and design algorithm. These models have multiple benefits, such as improving communication and collaboration in a development team via sharing models, increasing productivity by maximizing compatibility between systems through the reuse of standardized models, and supporting the early validation via models simulation.

After introducing the basic concepts of MBD, we teach the automated code generation and model-based V&V in details. Both are major factors that contribute to the efficiency improvement of the design, implementation and verification of safety-critical and security-critical embedded software systems.
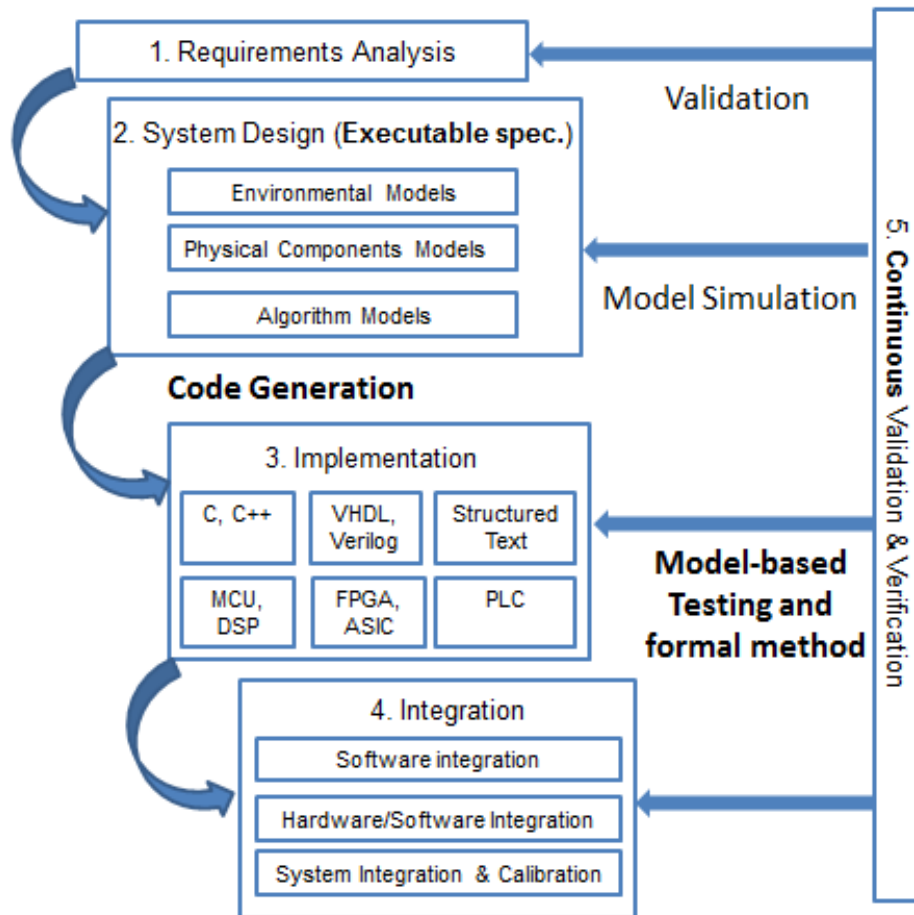
Figure 1: Model-based design process

Automated Ccode Generation

An increasing number of automated code generation tools have been created in the past decade. They help engineers to faster and better develop documented software in comparison to hand coded development. It has two outstanding advantages: (i) Eliminate errors from hand-coding; (ii) Regenerate easily for different targets. Many engineers with limited programming experience could be greatly relieved from low-level programming and focus, instead, on domain-specific issues. Moreover, depending on various application purposes, system design models can be translated to different implementation languages. For example, the program coded in Structured Text (a PLC language) is generated for PLC automation applications from Matlab/Simulink models; VHDL or Verilog code is generated for hardware specification models in FPGA or ASIC applications; for the microcontroller

control or DSP applications, models are translated to C/C++, which are dominant implementation languages. In addition, most high-level graphical programming languages like Matlab/Simulink and LabVIEW are easy to learn and use, but lack rigorous semantics. Some researchers have investigated the automated transformation from Simulink to a formal modeling language like Lustre, and applied existing formal verification tools to checking Lustre programs so as to formally verify Simulink models [8].

To give students a direct experience in automated code generation, the C code generated by a commercial tool Simulink Coder™ (formerly Real-Time Workshop®) and an open-source tool Gene-auto [9] have been exposed to students. First, the comparison of the code generated using these two tools is discussed in class. The C code generated from Simulink Coder is complex and hard to read.

This is mainly due to the additional code instrumented for performance optimization, code portability to various hardware targets, or debugging purposes. The resultant code can be used for both real-time and non-real-time applications, including simulation acceleration, rapid prototyping, etc. In contrast, the C code derived by Gene-auto is clean and easy to trace back to the corresponding Simulink model blocks. Such translated C code is mainly used for the program verification. Thus, the purpose is on functional correctness rather than execution performance. Next, the C code generated by the Gene-auto tool from Simulink design models is further studied for students to understand the details of the automated code generation mechanism.

Two translation examples are selected for the case study: (i) the translation of basic logical operation blocks, arithmetic operation blocks and simple subsystems composed of these two kinds of blocks to C functions or statements; (ii) the translation of states and transitions in Stateflow charts to C functions. For example, an addition "+" block with three inputs and one output can be translated to the C statement "o1 = in1 + in2 + in3". Students are asked to prepare a class report about the comparison of different code generation tools. Some relevant research papers were also offered to students who want to explore this topic further [10]. This topic not only helps students understand the automated code generation mechanism, but also convinces a student of the great enhancement achieved by the MBD approach to the system development by engineers.

Model-based Validation and Verification

Model-based V&V represents a set of verification and validation techniques *continuously* applied through the entire MBD process. It contributes to three important goals/benefits: (i) Detect design errors early in the development; (ii) Reuse the tests throughout the development process; (iii) Reduce the use of physical prototypes. In this course, three aspects

of model-based V&V techniques/tools are provided to students.

Firstly, conventional quality control techniques in software engineering [11] are recapped and compared. Validation is targeted at answering the question "Are we developing the *right* system?", while verification aims at answering a different question "Are we developing the system *right*?" Formal method and testing are two of the most popular approaches to hardware / software verification. In mission-critical systems, where bugs may occur with disastrous effects, formal methods are employed to guarantee the correct system behaviors with respect to the safety-critical requirements. In comparison, testing is scalable and easy to apply although it is limited to detecting bugs in a system; but it cannot ensure the correctness. Unit testing, integration testing and system testing are three common testing practices in the software systems development. The purpose of recapping quality control techniques is for students to clarify the typical usage differences of these techniques.

Secondly, V&V techniques applied at different MBD steps are discussed in class. During the initial requirement analysis step, a validator is applied to ensure that the extracted requirements correctly match the intended applications. In the system design step, a model simulator can be utilized to check whether the executable design specification satisfies the requirements obtained in the initial step. Unit testing is typically applied to check if the implementation coded in some low-level languages is functionally consistent with design models. Integration testing and system testing are initiated from the integration step. Formal methods are applied to check the critical components in the implementation. A translation validation tool, which formally verifies the translation from Simulink models to C, is also introduced to students.

Thirdly, latest advances of the model-based testing in both academics and industry are exposed to students. This is one of our new

teaching endeavors of integrating an on-going research project into advanced undergraduate level or graduate level courses; this has been reported in [12].

Eclipse-based Software Programming and Modeling Tools

Eclipse is a popular computer programming IDE originated from IBM VisualAge. More than eighty well-known IT companies (like QNX, Red Hat) besides IBM, have joined a foundation for the development and promotion of Eclipse. Many universities in the US currently adopt Eclipse for software programming and teaching computer science courses. Eclipse owns several unique characteristics for efficient and cost-effective computing system development. Firstly, by means of various plugins, developers can customize the Eclipse environment to develop applications using not only Java but also other high-level programming languages. For example, the C/C++ development tool (CDT) is an integrated development environment based on Eclipse to support developing applications in C/C++. Users can develop their own plug-ins to customize capabilities of the Eclipse platform. For example, some software model checkers include a graphical user interface, which is implemented as an Eclipse plugin for user-friendly verification and debugging. Secondly, as Eclipse itself is mostly implemented in Java (whose important feature is portability), it is

Table 1: Eclipse-based software tools for embedded design.

Name	Vendor	Major Apps	Features Description
CooCoxCoIDE	Open platform	MCU Programming Embedded real-time systems development	It consists of a component-based network platform and an eclipse-based IDE. It is configured with ARM GCC compiler and debugger for the ARM Cortex MCU based microcontroller programming.
Momentics Tool Suite	QNX	C/C++ programming, Embedded real-time systems development	A comprehensive IDE with profiling tools for analyzing system behavior such as real-time interactions, memory accesses.
eTrice	Eclipse	MBD for embedded systems, especially auto-code generation	An Eclipse project for embedded model driven software development using a special system model called ROOM. It supports the code generation in Java, or C/C++.
TOPCASED	Airbus	MBD for embedded systems, especially auto-code generation, formal verification	An Eclipse based software environment dedicated to the realization of critical embedded systems. It supports formal checking and code generation from models in sysML or UML to Java, C or Python.
Code Composer Studio	Texas Instruments	MCUProgramming	An IDE for developing applications for TI embedded microprocessors, including DSPs, ARM based MCUs. It includes a real-time operating system, so as to support OS level application debug and low-level JTAG based software development.
Xilinx Platform Studio	Xilinx	MBD for FPGA based embedded system development	The embedded development kit for building MicroBlaze embedded processor systems in Xilinx FPGAs. One tool Xilinx Platform Studio in this kit allows designers to configure the hardware specification of the system and automatically converts the specification into a RTL description using VHDL or Verilog.

cross-platform and runs on any hardware/operating-system platform, like Linux and Mac OS besides Windows. This feature is particularly useful for developing Linux-based embedded systems. Thirdly, Eclipse is free and open source software, and other Eclipse-based development tools that are released under the Eclipse Public License are also free.

Besides the flexible, portable and cost-effective programming environments that Eclipse provides, Eclipse is also one of the model-based design initiatives. Eclipse modeling framework (EMF) provides the modeling and code generation facility for building tools and other applications based on a formatted data model [13]. From a model specification described in XML Metadata Interchange (XMI), EMF supports constructing software tools to automatically create the code implementation for the design model, a set of utility classes enabling editing the models and a basic editor. There are a large number of EMF-based software tools available for multiple MBD usages, such as composite modeling, model transformation, model simulation and checking, and code synthesis.

As this PT course is for engineering and technology students, the emphasis of this course is on introducing the EMF-based tools for the MBD of embedded systems, instead of the working mechanism of EMF itself. In this course, we first briefly discuss the Eclipse platform including its origin and architecture, especially its innovative plug-in extensions and support for multiple programming languages. Some popular Eclipse plugins are exposed to students, such as Eclipse Subversive called Subversion for version control.

Recently, Eclipse-based development tools have become available for dealing with a wide range of embedded systems development problems. Table 1 shows a list of Eclipse-based software tools for embedded systems related applications. We have selected three tools to introduce to students in this course: Code Composer Studio (CCS) from Texas Instruments (TI) – an IDE for MCU programming; TOPCASED – a development environment dedicated to the modeling, code synthesis and verification of safety critical embedded systems; and Xilinx Platform Studio – a software tool for developing FPGA based system-on-chip electronic hardware. The primary experience of teaching these tools is presented in the following.

TI/CCS: To teach students using CCS for MCU programming, we choose the TI TM4C1294 Connected Launch Pad as the hardware board for developing a MCU-based embedded system in this course. The MCU included in this board is the 32-bit ARM Cortex-M4. A step-by-step tutorial on using CCS for the MCU programming has been prepared, starting from launching CCS, importing or creating a C/C++ project, configuring the project compilation and linking properties, building and debugging a project to loading the built project to the flash memory of the target board. After getting familiar with CCS, students have developed a capstone project for the *Internet of Things* application under this environment: a web-based remote temperature monitoring and control system for home automation.

TOPCASED: The main motivation of introducing this tool is to help students be aware of formal verification of design models in this environment. We introduce students to multiple simple demo examples instead of teaching the formal semantics of the back-end TOPCASED modeling language, which is too overwhelming for non-CS students. The transformation from the front-end modeling language AADL to the back-end formal language and the underlying model checker used for the automated formal verification are also provided to students.

Xilinx platform studio: As the capacity of FPGA devices grows rapidly, a System on a chip (SoC) can be realized in a programming chip. A lab manual has been prepared to teach students utilizing this software tool to design the SoC based FPGA. By following the manual, all students have successfully used this software

tool to develop the hardware design projects, especially customizing the SoC by configuring the Xilinx MicroBlaze software core and selecting memory modules and IO modules. This tool also enables the automatic generation of the hardware design in Verilog HDL.

These tools can help students design and implement their senior course projects more efficiently. More importantly, all Eclipse-based tools share common features from project management, code/model edit view, project building and execution, and debugging perspectives. After students grasp Eclipse and one or two Eclipse-based tools, it is easy for them to learn and use other new Eclipse-based tools, which reside in the embedded systems development.

Conclusion

MBD is an advanced and cost-effective development methodology for developing complex and safety-critical embedded software systems. This paper presents our teaching experience of integrating this new MBD paradigm into a system-level Programming Tools course for electrical and computer engineering and technology students. It mainly describes two new topics integrated in this PT course: MBD and eclipse-based software tools, from course materials preparation and instruction approaches to the two aspects. Eclipse and other Eclipse-based tools are observed to be useful in more than just software programming instruction in computer science courses. Our experiences show that such tools in combination with MBD are also helpful in embedded computer systems education for engineering and technology students. In the future, our students and our teachers will together create more capstone projects utilizing the MBD paradigm in combination with Eclipse-based software tools. As we collect more data, we hope to objectively verify the educational value of incorporating into our appropriate courses Eclipse-based tools in combination with model-based design methodology.

References

1. Link to US bureau of Labor Statistics: http://www.bls.gov/emp/ep_table_102.htm,

2. EU MOGENTES project, http://www.mogentes.eu/, 2014.

3. SESAME project http://wiki.lassy.uni.lu/projects/SESAME

4. J. L. F Aleman, "Automated Assessment in a Programming Tools Course," *Education, IEEE Transactions on*, vol.54, no.4, pp.576-581, Nov. 2011.

5. Paul G. Flikkema. "Approaching the Design of Complex Engineered Systems: A Model-based Approach Informed by System Thinking". *Proceedings of ASEE PSW Conference*, 2012.

6. P. J. Clarke, Y. Wu, A. Allen, and T.M. King, "Experiences of Teaching Model-driven Engineering in a Software Design Course", *ACM/IEEE Intl. conference on Model Driven Engineering Language and Systems*, Oct. 2009.

7. Mireille Blay-Fornarino. *"Project-based teaching for Model-Driven Engineering"*, in *Proceedings of Promoting Software Modeling through Active Education*, pages 69-75, Sept 2008.

8. A. H. Joshi, "Model-based safety analysis of Simulink models using SCADE design verifier". *Proceedings of Computer Safety, Reliability, and Security,* vol. 3688, Springer, 2005.

9. Gene-auto project. http://geneauto.gforge. enseeiht.fr/

10. C. Silva, S. Correa, A.M. Cunha, V. Dias, and O. Saotome, "A comparison between automated generated code tools

using model based development," *Proceedings of Digital Avionics Systems Conference,* pp.1-9, Oct. 2011.

11. S. Bran, "The Pragmatics of Model-driven Development", *Software, IEEE*, vol.20, no.5, pp.19-25, 2003.

12. Nannan He. "Incorporating On-going Verification & Validation Research to a Reliable Real-Time Embedded Systems Course". *Proceeding of the ASEE North Midwest Sectional Conference*, pp.402-408, Fargo, Oct. 2013.

13. Eclipse modeling project. http://www.eclipse.org/modeling/

Biographical Information

Nannan He is an Assistant Professor of the ECET Department in Minnesota State University at Mankato, Minnesota, USA. She received the Ph.D. in computer engineering from Virginia Tech. Her teaching and research interests are in safety-critical embedded software, real-time systems, and software verification.

Han-Way Huang is a Professor of the ECET Department of Minnesota State University at Mankato, Minnesota, USA. He received the Ph.D. in computer engineering from Iowa State University