Utilization of Eclipse-based Software Tools in Teaching a New Software Development Methodology to Engineers

Dr. Nannan He, Minnesota State University, Mankato

Nannan He received the Ph.D. in computer engineering from Virginia Tech. She did Post-doc at Oxford University in UK and participated two EU projects. From 2012 to present she is an Assistant Professor at the ECET department in Minnesota State University at Mankato. Her teaching and research interests are in safety-critical embedded software, real-time embedded systems, and software verification. She is an IEEE member and reviewers for many conferences and journals in EDA field.

Dr. Han-Way Huang, Minnesota State University, Mankato
Dr. Nannan He, Minnesota State University, Mankato

Nannan He received the Ph.D. in computer engineering from Virginia Tech. She did Post-doc at Oxford University in UK and participated two EU projects. From 2012 to present she is an Assistant Professor at the ECET department in Minnesota State University at Mankato. Her teaching and research interests are in safety-critical embedded software, real-time embedded systems, and software verification. She is an IEEE member and reviewers for many conferences and journals in EDA field.

Utilization of Eclipse-based Software Tools in Teaching a New Software Development Methodology to Engineers

Abstract

Software development is often considered to be difficult for engineering students. Nowadays, in many embedded systems, software portion is always expected to have the greater impact on the behavior of entire systems. Therefore, educators continue to face great challenges in getting students to be capable of conducting efficient software development. This paper presents our experiences of introducing both eclipse-based tools and advanced 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. Recently, eclipse-based development tools have been employed by increasing number of software projects in both academy and industry. Many eclipse-based software tools support MBD, which is an emerging development methodology for complex embedded software. The novelty of our work is to introduce students the MBD process in combination with eclipse-based tools. The goal is to equip engineering students with the knowledge of using real-world software tools and cost-efficient software development methods. Our primary observations show that this combination could help students understand advanced software development technologies in practice, and improve the efficiency of designing and implementing complex embedded software projects.

1. Introduction

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 students. Engineering students usually study the syntax and semantics of low-level programming languages (PL) such as C or assembly in one or two semesters. They have fewer opportunities to apply programming skills compared with computer science major or software engineering major students. It is common to forget the syntax of C language among engineering students. When a class project involves software programming, students often spent a large amount of time in debugging syntax and semantics errors, with little time left for algorithm development and verification. Many engineering students consider writing a small program with 300 to 500 lines of code as a painful

experience. And a large percentage of junior or senior design projects that could not be accomplished on time are due to the prolonged software implementation stage.

Model-based design (MBD) is an emerging methodology for developing complex software, especially embedded software. Its efficiency has been demonstrated in software engineering. For example, the Matlab/Simulink language from MathWorks that supports 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 the software system is either generated or derived manually from high-level models. In recent years, multiple large EU-funded research projects have been launched or completed to promote the application of MBD in industry, and target at solving challenges encountered in different real-world application domains, such as CESEAR project [2] (Cost-efficient methods and processes for safety relevant embedded systems), MOGENTS project [3] (Model-based Generation of Tests for Dependable Embedded Systems), and SESAME project (A Model-driven Test Selection Process for Safety-critical Embedded Systems) [4]. However, there are few universities in America that offer engineering students the knowledge of MBD.

In the last decade, eclipse-based tools and MBD methodology have been widely applied to developing dependable embedded software systems in various embedded applications such as automobile and automation. Eclipse is an integrated software development environment originated by IBM, which comprises a base workspace and an extensible plug-in system for customizing the environment. Eclipse owns several important benefits for computer programming. First, by means of various plugins, Eclipse provides the multi-language programming, such as Java, C/C++. Second, Eclipse is cross-platform, which supports Windows, Linux and Mac OS as well. Thus, it is preferable for developing Linux-based embedded systems. Third, Eclipse is free under the term of the Eclipse Public License. Most Eclipse-based software tools are also free for the education purpose. In recent years, eclipse-based development tools have been employed by more and more software projects in both academy and industry. These tools have become available for a wide range of the embedded systems development problems, such as microcontroller programming, systems modeling for embedded model-based design, real-time computing platform, and FPGA based embedded system development. More importantly, Eclipse modeling framework (EMF) which is an eclipse-based modeling platform and code generation facility, is one of the most popular and well-know MBD initiatives.

2. Programming tools course description

The course Programming Tools (or with the similar name) is usually required for computer engineering or computer science major students, but an elective course for other engineering

major students in many universities in US. It can be offered at 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 [5]. At the systems level, it typically focuses on key concepts of system-level programming (e.g., C/C++, Python and LabVIEW input language); tool chains for group software development; and advanced topics on software system design, implementation, testing strategies and documentation [6].

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 capable of utilizing existing 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 created system often deviates from the original design plan. Educators have recognized the need to introduce some efficient and cost-effective programming tools to students [7]. The main goal is to equip students with the knowledge for developing complex engineering systems with a large number of constraints.

Experts in the software engineering and computer science communities advocate introducing the MBD methodology to students. It provides students with insights, techniques and tools to alleviate difficulties of developing complex software systems. Educators have either integrated MBD into the existing software design course [8] or proposed a new project-based course to solely teach MBD [9]. However, as these courses are mainly for computer science students, their contents are too theoretical for engineering students who have limited software development background.

The intent of our PT course is to convey the practical knowledge related to programming to students. We added materials on MBD from the engineering practitioner's point of view to this course with three objectives.

- To improve students' awareness of the advanced MBD methodology.
- To develop students' appreciation for MBD that will contribute to the efficient and cost-effective application development.
- To give students the opportunity to learn modern programming tools enabling MBD.

Two unique features of MBD: automated code generation and model-based verification and validation besides the basic MBD concept have been integrated into the PT course.

Model-based Design Concepts

We introduced key MBD concepts that are important for an engineering practitioner to our students during the first week. Five basic steps in MBD approach from requirement analysis,

system design, implementation, integration to continuous verification, were covered. Based on the MBD process illustrated in Figure 1, we discussed main differences between MBD and conventional software development processes like the waterfall model to encourage active learning. For instance, without being taught, students can summarize by themselves that testing and verification is conducted continuously during each of the other four steps, not until their completion. Students are also guided to learn new concepts along each basic MBD step. Using System Design step as an example, students learn the concept of Executable Specification (in terms of models shown in Figure 1). It can unambiguously model the entire system functionality, including the environment, physical component and design algorithm. These models have multiple benefits, such as improving communication and collaboration in the development team via sharing models, increasing productivity by maximizing compatibility between systems through the reuse of standardized models, and supporting early validation and testing via models simulation.

After introducing basic concepts of MBD, we taught 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.

Automated code generation

Increasing number of automated code generation tools has been created in past ten years. This is mainly because they help engineers 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 on domain-specific problems. Moreover, depending on application purposes, system design models are synthesized to different implementation languages. For example, programs coded in the Structured Text (a PLC language) are generated for PLC automation applications; VHDL or Verilog code is generated for hardware specification models in FPGA or ASIC applications; for MCU control or

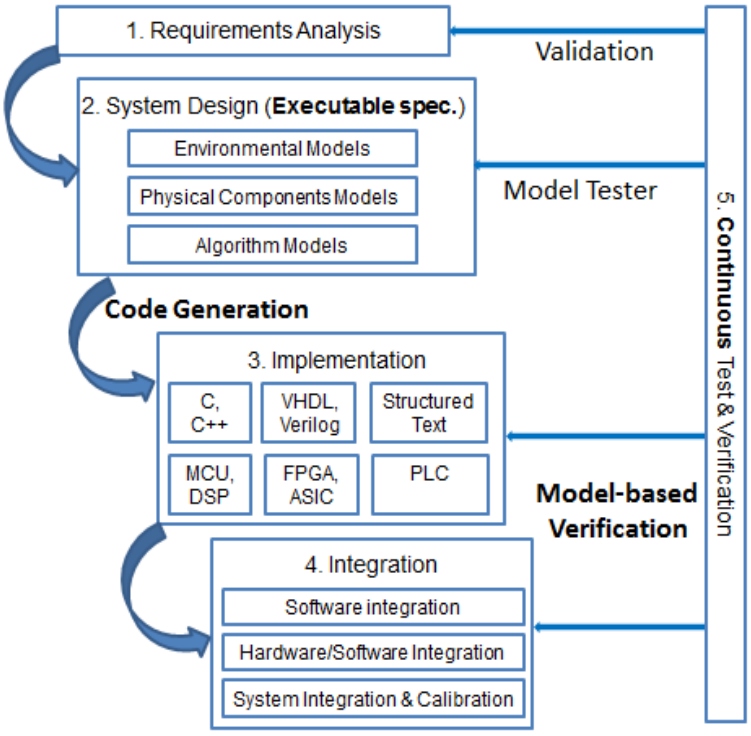

Figure 1 Main steps in Model-based design process

DSP applications, models are translated to C/C++, which are dominant programming languages in these applications.

Formal verification of specification models aims at ensuring the correctness of the design. Most high-level programming languages like Matlab/Simulink are easy to use, but lack of rigorous semantics. Some researchers have investigated the automated transformation from Simulink to a formal language like Lustre, and applied existing formal verification tools for checking Lustre programs so as to formally verifying Simulink models [10].

To give students a direct experience of automated code generation, the C code generated by a commercial tool Simulink Coder™ (formerly Real-Time Workshop®) and an open-source tool Gene-auto [11] were exposed to students. First, the comparison of the generated code 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 injected into the C Code for the performance optimization or debugging purposes. The resultant code can be used for real-time and non-real-time applications, including simulation acceleration, rapid prototyping, etc. In contrast, the C code derived by Gene-auto could be clean and easy to trace back to the corresponding Simulink model blocks. The C code translated by Gene-auto is mainly for program verification. Thus, the focus 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 details of automated code generation mechanism.

Two translation examples are selected for 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 could be translated to the C statement "o1 = in1 + in2 + in3". Students are asked to prepare a class report on the comparison of different code generation tools. Some relevant research papers were also offered to students who want to explore this topic further [12]. This topic not only helps students understand automated code generation mechanism, but also convinces student great enhancements led by the MBD approach in system development for engineers.

Model-based validation and verification (V&V)

Model-based V&V represents a set of V&V techniques continuously applied through the MBD prcess. All of them contribute to three important goals/benefits: (i) Detect errors early in the development; (ii) Reuse test throughout development process. (iii) Reduce use of physical prototypes. In this course, model-based V&V techniques/tools in three aspects were provided to students.

First, conventional quality control techniques in software engineering [13, 14] were recapped and compared. Validation targets 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 most popular approaches for hardware / software verification. In mission-critical systems, where bugs may incur disastrous effects, formal methods are employed to guarantee the correct behavior with respect to the safety-critical requirement. In comparison, testing is scalable and easy to apply although it is limited to detect bugs in a system, but cannot ensure the correctness. Unit testing, integration testing and system testing are three common testing practices in software systems development. The purpose of recapping quality control techniques is for students to clarify the typical usage and differences of these techniques.

Second, 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 extracted requirements correctly match the intended use. In the system design step, a model tester or a simulator can be utilized to check whether the Executable Specification satisfies requirements obtained in the initial step. Unit testing is typically applied to check if the implementation coded in some low-level languages is consistent with design models. Integration testing and system testing are initiated from the integration step. Formal methods are applied to check critical components in the implementation. A translation validation tool, which formally verifies the translation from Simulink models to C, is introduced to students [*16*].

Third, latest advances of model-based testing in both academy and industry are exposed to students. This is one of our new teaching endeavors in integrating an on-going research results into advanced level or graduate level courses.

3. 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 US currently also use Eclipse for software programming and designing embedded systems courses. Eclipse owns several unique characteristics for the efficient and cost-effective computing system development. First, by means of various plugins, developers could customize environments of Eclipse to develop applications in 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 the Eclipse to support developing applications in C/C++. Users can develop their own plug-ins to customize capabilities of Eclipse platform. For example, some software model checkers include a graphical user interface, which is realized as an Eclipse plugin for the user-friendly verification and debugging [17]. Second, as Eclipse itself is mostly implemented in Java whose important feature is portability, it is cross-platform and can run on any hardware / operating-system platform, like

Linux and Mac OS besides Windows. This feature is very helpful for developing Linux-based embedded systems. Thirdly, Eclipse is free and open source software, and Eclipse-based development tools[18] that are released under the Eclipse Public License are also free.

Name	Vendor	Major Applications	Features Description
CooCox CoIDE	Open platform	Programming microcontroller, 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. It also supports component oriented programming. http://www.coocox.org/coocox_coide.htm
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. http://www.qnx.com/products/tools/qnx-momentics.html
eTrice	Eclipse	MBD for embedded systems, especially modeling and 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++. http://www.eclipse.org/etrice/
TOPCASED	Airbus	MBD for embedded systems, especially modeling, 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. http://www.topcased.org/
Code Composer Studio	Texas Instruments	Programming microcontroller	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. http://www.ti.com/tool/ccstudio
Xilinx's EDK (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. http://www.xilinx.com/tools/platform.htm
EZRealtime	UFAM/EMF	MBD for developing embedded and real-time systems	A MBD-based tool based on timed Petri Net formalism for developing embedded and real-time systems. http://code.google.com/p/ezrealtime/

Table 1. A list of Eclipse-based Software Tools

Besides the flexible, portable cost-effective *programming* environments that Eclipse provides, Eclipse is also one of the most well-known 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. From a model specification described in XMI, EMF supports constructing tools to automatically produce the code implementation for the model, a set of utility classes that enable editing the model and a basic editor. There are a large number of EMF-based software tools available for multiple MBD purposes, such as composite modeling, model transformation, model simulation and checking, and code synthesis[19].

As this PT course is for engineering students, the emphasis of this course is on the introduction of 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 to students, especially its innovative plug-in extensions and their support for multiple programming languages. Some popular Eclipse plugins are exposed to students, such as Subversive for the version control.

Recently, Eclipse-based development tools have become available for dealing with a wide range of the embedded systems development problems, such as microcontroller programming, systems modeling, real-time computing platform, and FPGA based embedded system development. Table 1 shows a list of Eclipse-based software tools for these embedded systems related applications. We selected three tools to introduce to students: Code Composer Studio from TI – an eclipse-based IDE for microcontroller programming; TOPCASED – an eclipse-based software environment dedicated to the modeling, code synthesis and verification of safety critical embedded systems.

4. Eclipse-based embedded software development education in China

Embedded systems development has become one the key discipline directions in many universities and research centers in China. The investigation on the Eclipse based development environment and tools specific for embedded system analysis and design has started in 2009 [20]. The focus was on the development tool of C/C++ such as CDT. IBM China plays an important role in promoting Eclipse-based tools. In the last two to three years, the study on Eclipse-based software tools has gradually extended to the design of new educational modules for assisting eclipse-based programming courses, for example, some eclipse plugins have been developed for advising and automatic checking and correction of programming assignments [21]. Around 10-15 top universities in China have provided the Eclipse related courses in their embedded system major programs. According to the Embedded China conference in 2013, it is expected in the next 2-3 years more embedded system developer engineers in China will choose Eclipse-based tools and be able to customize new tools to perform complex embedded systems development. It

is worthwhile noting that the trustworthy embedded system design is gradually attracting more researchers to exploring. MBD and MBT is one the major research directions. Chinese Natural Science Foundation has supported multiple large scale projects on MBD of trustworthy embedded systems since 2012. However, few universities in China offer courses on Eclipse-based MBD of embedded system, only the graduate school in Chinese Science Academy has opened a research course recently as far as best knowledge.

5. Conclusion

MBD is cost effective for developing complex and reliable-critical embedded systems. This paper presents our teaching experiences of integrating this new MBD paradigm into a system-level Programming Tools course for CE and EE students. It mainly describes two new topics integrated to this PT course: MBD concepts and eclipse-based software tools supporting MBD, from the course materials preparation and instruction approaches two aspects. In the future, students and our teachers will together create and gather more capstone projects related to MBD by means of eclipse-based software tools.

References

1. Link to US bureau of Labor Statistics: http://www.bls.gov/emp/ep_table_102.htm, a related Link to the market for computing careers: http://cs.calvin.edu/p/ComputingCareersMarket
2. EU CESAR project (Cost-Efficient Methods and Processors for Safety Relevant Embedded Systems) http://www.cesarproject.eu/
3. EU MOGENTES project (Model-based Generation of Tests for Dependable Embedded Systems) http://www.mogentes.eu/
4. SESAME project (A Model-driven Test Selection Process for Safety-critical Embedded Systems) http://wiki.lassy.uni.lu/projects/SESAME
5. Aleman, J.L.F., "Automated Assessment in a Programming Tools Course," *Education, IEEE Transactions on*, vol.54, no.4, pp.576-581, Nov. 2011.
6. Links to some system-level PT courses: http://www.cs.washington.edu/education/courses/cse374/; http://web.eecs.utk.edu/~huangj/cs360/ http://school.eecs.wsu.edu/undergraduate/cpts/courses/360
7. Paul G. Flikkema. "Approaching the Design of Complex Engineered Systems: A Model-based Approach Informed by System Thinking". *Proceedings of ASEE PSW Conference*, 2012.
8. Peter J. Clarke, Yali Wu, Andrew A. Allen, and Tariq 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.
9. Mireille Blay-Fornarino. "Project-based teaching for Model-Driven Engineering", in *Proceedings of the Promoting Software Modeling through Active Education*, pages 69-75, Sept 2008.
10. Joshi, A., Heimdahl, "Model-based safety analysis of Simulink models using SCADE design verifier". *Proceedings of Computer Safety, Reliability, and Security (SAFE-COMP)*. Volume 3688 of LNCS, Springer (2005).
11. Gene-auto project. http://geneauto.gforge.enseeiht.fr/

12. Coelho da Silva Stanisce Correa, G., da Cunha, A.M., Vieira Dias, L.A., Saotome, O., "A comparison between automated generated code tools using model based development," *Digital Avionics Systems Conference (DASC), 2011 IEEE/AIAA 30th* , vol., no., pp.7E4-1-9, Oct. 2011.
13. G. J. Myers, C. Sandler, T. Badgett, and T. M. Thomas, "*The Art of Software Engineering*", Hoboken, NJ:Wiley, 2004.
14. R. Pressman, "*Software Engineering: A Practitioner's Approach*", New York: McGraw-Hill, 2009.
15. Bran Selic, "The Pragmatics of Model-driven Development", *Software, IEEE*, vol.20, no.5, pp.19-25, Sept.-Oct. 2003.
16. O. Strichman, M. Ryabtsev, "Translation validation: from Simulink to C", *Proceeding of Intl' Computer Aided Verification conference*, pp 696-701, 2009.
17. The CProver User manual: http://www.cprover.org/satabs/download/manual.pdf
18. Link to Eclipse based software: http://en.wikipedia.org/wiki/List_of_Eclipse-based_software
19. Link to Eclipse Modeling Framework based software: http://en.wikipedia.org/wiki/List_of_Eclipse_Modeling_Framework_based_software
20. Nan Fang, "Eclipse-based Integrated embedded development environment analysis and design", Master Thesis in Xi An electronics technology university, 2009.
21. Zhao Kang, "Research and design and Eclipse-based programming modules for the experimental education systems", Master thesis, Beijing University of Posts and Telecommunications, 2013.