Use of Model-based Design to Teach Embedded Systems Programming

Nannan He and Han-way Huang
Dept. of Electrical and Computer Engineering Technology, Minnesota State University at Mankato
nannan.he@mnsu.edu and han-way.huang@mnsu.edu

***Abstract*— As embedded systems become increasingly complex, there is a great need to equip engineering students with the knowledge of advanced embedded software development techniques so as to improve their efficiency in software development and programming. paper presents our experiences of introducing the model-based design (MBD) methodology to two computer engineering related undergraduate courses: Programming Tools and Real-time Embedded Systems. MBD is an emerging design methodology whose effectiveness and efficiency have been demonstrated in the development of embedded software systems in industry. Matlab/Simulink from MathWorks supports the MBD and has become the predominant software modeling language in many safety-critical embedded applications. The novelty of this teaching effort is incorporating two on-going research projects on the MBD with the Matlab/Simulink into education. The synergistic benefits of integrating research with teaching have been explored in two different manners. Our experiences demonstrate that the integration of research results with educational practices is efficient to improve the effectiveness of teaching programming and software development to engineering students, and promote undergraduates to participate in research projects as well.**

Keywords— model-based design; Matlab/Simulink; embedded programming; verification; research; microcontroller

I. INTRODUCTION

Nowadays, embedded computing system plays an important role in many safety- and reliability- critical applications, such as automobile, railways, aircraft domains. Thus, the correctness of embedded software must be rigorously validated and verified before it is put into operation. At the same time, as the computing power of the microprocessor and the complexity of peripheral devices greatly increase, such as gyro sensors, Ethernet port and WiFi extension board for networking, embedded systems for controlling these devices become much more complicated. Therefore, there is a great need to equip students with the knowledge of advanced embedded software design techniques so as to improve their efficiency in developing reliable embedded systems. Moreover, software development training could be a valuable experience for students, as it can cultivate students' problem solving and process development capability.

However, software programming is often considered to be challenging for engineering and technology students. Many students have struggled with minor programming assignments, and do not know how to program well at the end of the introductory programming courses taught with the conventional lecture format [1, 2]. It is common for engineering students to forget the syntax of C language when they work on the junior or senior design projects, and a large percentage of these projects that cannot be accomplished on time are mainly due to the prolonged software implementation stage.

Model-based design (MBD) is an emerging methodology for developing complex software, especially in safety-critical embedded systems. By promoting the use of domain-specific notations to *graphically* represent designs specifications and the continuous model-based verification and validation (V&V), the MBD can identify design flaws at the early stage. Recently, multiple large EU-funded research projects have been launched to promote the wide application of MBD in industry, and to solve challenges encountered in various real-world application domains [3, 4]. More system modeling tools have started supporting the MBD, such as the Simulink software package from MathWorks. In fact, MathWorks organized a showcase presentation in 2012 and a workshop in 2014 in this conference to present this tool for teaching courses closely related to programming. Matlab/Simulink allows users to directly program a wide range of low-cost, easy-to-use hardware boards, like Ardunio, Raspberry Pi, etc. These events inspired us to introduce the MBD by using the Matlab/Simulink to teach embedded systems programming in computer engineering curriculum.

At the same time, there are increased demands on undergraduate research. Firstly, high technology companies need to hire employees with research background to keep companies competitive. Secondly, graduate schools are more interested in enrolling students already equipped with certain research skills. Opportunities for undergraduates to participate in researches have increased recently due to a variety of federally or industrially funded programs, such as Experiences for Undergraduates (REUs) funded by the National Science Foundation [5, 6], and Minnesota Center for Excellence in Manufacturing & Engineering [7]. The work reported in this paper aims at encouraging undergraduate engineering students to engage in practical research. The main objective is to provide students with quality research experiences in engineering, especially advanced embedded software design methodologies like MBD and software verification for the safety-critical embedded systems construction.

Two related on-going research projects conducted by faculties have been incorporated into this teaching practice. One project called MVT is on the model-based verification and testing of Simulink models. An automated formal method

called model checking [8] has been combined with mutation testing to generate from Simulink design models high coverage test suites, which are used to accurately and efficiently verify software implementation. Another project called MBD-MCU is on developing microcontroller (MCU)-based applications by applying the MBD. A set of custom Simulink blocks has been developed to support directly programming various hardware platforms for Internet of Things (IoT) applications without writing the C code. The hardware platform currently used is Arduino Uno. The benefits of integrating research with teaching have been explored in two ways. Firstly, the research methods and results of two research projects are exposed to students in the form of course projects. These projects give students rich hands-on experiences of efficiently developing, verifying and analyzing embedded software by utilizing existing software tools that enable the MBD in practice. Overall, these new teaching materials derived from research enrich the courses' contents and equip students with not only the MBD concepts but also the practical knowledge of applying the modern MBD. Secondly, these teaching materials inspire students to investigate new advances in the frontier of the MBD. Two students got undergraduate research grants on the MBD afterwards to investigate this topic further. Such results demonstrate that students' software programming efficiency and confidence have improved.

This paper starts with the related work and background of MBD, followed with basic course information, student evaluation and feedbacks, and concludes with some thoughts on future work.

II. Related Work

The complexity and importance of learning programming for undergraduate students have generated an active educational research area [9-11]. The results of analyzing common programming bugs by computer science students show that about 22% of the problems are due to serious problem solving limitations [9]. Some investigation indicated that a flow-based visual programming language or even a natural language could address some of the most difficult topics to learn programming as students do not need to learn a formal grammar [10, 11]. In our work, Simulink as a graphical dataflow language is a promising and attractive option to teach students the embedded programming by focusing on improving their problem solving ability. Introducing testing and verification in teaching programming haven been recently reported. The findings in [12] reveal that the incorporation of mutation testing in programming courses adds valuable knowledge to the learning process. [13] presents an approach to enhancing students' competency in software formal verification. Both model-based testing and model checking which are important V&V techniques of embedded software have been introduced to students in this work.

III. Background

The MBD process typically consists of five major phases as illustrated in Figure 1. It has three distinguished features compared with the conventional waterfall or V model: a) executable design specification which supports the early V&V via either models simulation or formal technique like model checking. b) automatic code generation with two outstanding advantages - eliminating errors from hand-coding and regenerating code easily for different application targets. c) continuous model-based V&V, which represents a set of V&V techniques *continuously* applied through the process, from the requirement validation, model simulation and model checking to model-based testing on the implementation.

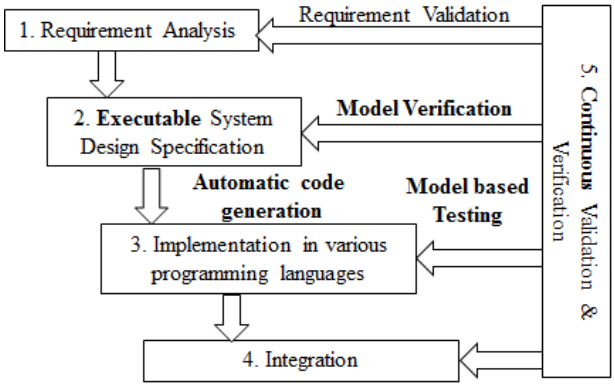

Figure 1. Model-based design process

Matlab/Simulink from MathWorks supports MBD and has become the predominant software modeling language in many embedded applications. It is a graphical dataflow language used to unambiguously model the dynamic systems functionality and enable the multi-domain simulation. A Simulink diagram consists of a set of blocks connected by signals which specify the flow of data. Blocks are instantiated by pre-defined blocks, which are grouped into different libraries according their functionality. Common block libraries include *Math, Logic and Bit, Sinks, Sources, etc*. The example diagram in Figure 2 includes a *Pulse Generator* block as *source* block to generate signals at the specified frequency, a Digital Output block as *sink* block and the directed line which passes signal from the source to the sink. This output block represents the general purpose input/output pin 9 on the Arduino Uno board.

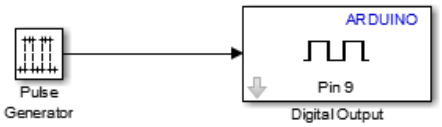

Figure 2. Example Simulink design diagram

MathWorks recently supported connecting Matlab and Simulink to directly program hardware. Some are for the engineering education and small prototyping such as Arduino, Raspberry Pi [14]. For example, three Simulink blocks libraries for programming the Arduino are *Common, Ethernet Shield* and *Wifi Shield*. Each library currently includes 5 to 10 blocks which enable users to perform basic microcontroller tasks, like acquiring analog and digital sensor data, controlling other devices with digital and Pulse Width Modulation (PWM) outputs, etc. Moreover, users can create new custom Arduino Simulink blocks to enhance existing libraries.

IV. Basic Courses Information

Teaching materials on MBD have been included in two courses: Programming Tools (PT) and Real-time Embedded Systems (RTES). The two courses have some common learning objectives on this new subject. However, the PT course focus on applying the MBD to efficiently developing reliable the MCU-based software systems; the RTES course emphasizes the model-based V&V to check the correctness of embedded software. The student outcomes on the subject of MBD in two courses are given in Table 1.

Table 1. Student outcomes

Student outcomes
O1. (both PT and RTES) Can demonstrate the basic comprehension of the MBD methodology (i.e., the workflow of the MBD and its three unique features)
O2. (PT only) Can apply the MBD concepts to programming MCUs by using a graphical programming language to construct executable design models for IoT applications (i.e. Matlab or LabVIEW)
O3. (RTES only) Can apply the model-based V&V techniques and tools to test and formally check the correctness of embedded software.

The teaching of MBD techniques in both courses is organized in a way that the part of the classes are spent on covering basic concepts and design principles, but most class time is used to work on projects either individually or in the 2 to 3 students group. Table 2 presents the week-by-week topics covered in both courses. Two courses share the same topics in the first two weeks, which serve the student outcome O1. Starting from the week 3, the topics and projects are used to achieve the outcome O2 and O3 separately.

Table 2. Course topics organization

week	**Topics**
1	(PT and RTES) Basic MBD workflow, key features of the MBD (i.e., executable design models, automated code generation, and continuous model-based V&V); research opportunities in the MBD.
2	(PT and RTES) Comparison of the MBD with conventional software development methodologies; overview of common software tools supporting the MBD (i.e., Matlab/Simulink or LabVIEW) and case studies.
3	(PT) basic syntax of graphic dataflow input programming language of Matlab/Simulink package; commonly used Simulink blocks libraries; customize Simulink blocks; automatic code generation tool.
	(RTES) Basic V&V techniques; formal methods and testing techniques, verification tools (i.e., mutation testing, model checking etc.)
4-5	(PT) Projects (given in Table 3) on programming the MCUs by applying the MBD with the Simulink package.
	(RTES) Projects (given in Table 4) on the model-based verification and testing by applying the bounded model checking and mutation-based testing techniques.

A series of projects designed for the *PT* course is derived from the on-going research project MBD-MCU, as shown in Table 3. All these projects make use of the Arduino Uno as the underlying MCU and the Ethernet and WiFi shields as hardware extensions. Project-based learning (PBL) has demonstrated a powerful learning strategy which engages students and creates an association between theory and practice by teaching concepts through real world problems [15]. All projects are designed to be open-ended, with the goal of developing the MCU based IoT applications by using Simulink libraries and hardware support packages. Students are required to complete project 1 to design a Simulink diagram to perform basic MCU functions involving general purpose inputs and outputs with peripheral devices, such as digital/analog inputs and outputs, PWM waveform generation for motor control etc. From project 2 to 4, students need to use existing Simulink blocks to develop the various machine to machine communication functions: common serial communications, wired or wireless networking connectivity. In project 5, students are required to design new customized Simulink blocks.

Table 3. Projects design and learning outcomes in *PT*

Project Description	**Learning outcomes**
1. Design a Simulink diagram to control the general purpose input and output of the MCU.	- Apply basic blocks from the Simulink support package to programming and testing the MCU applications.
2. Design a Simulink diagram for the serial communication (i.e. I2C and SPI) between MCUs.	- Configure the parameters of the blocks from the Simulink support package to program and test the MCU and its serial interfaces.
3. Develop a Simulink program for Ethernet applications (i.e. TCP/IP or UDP).	- Apply the advanced Simulink blocks for the MCU based wired networking applications.
4. Develop a project in Simulink with the WiFi connectivity for basic IoT applications.	- Apply blocks from both common Simulink libraries and support package for the specific hardware.
5. Develop new Simulink blocks using S-functions for the hardware.	- Create new Simulink blocks to enrich the support package for the specific hardware.

The projects for the *RTES* course are based on another research project called MVT funded by Minnesota State University at Mankato. Technically speaking, software verification checks that the system built after each development step satisfies the design specification. Testing and formal methods are two popular software verification approaches. In the context of model-based testing, traditional code-based metrics are no longer sufficient to estimate the quality of software testing. Thus, the mutation-based coverage metric which is a fault-based coverage is utilized in this research. On the other hand, model checking as an automated formal method is particularly well suited in the MBD paradigm. Its idea is to exhaustively examine every possible combination of system input and state, and prove or disprove design properties. In case that the property does not hold, the counterexample that demonstrates the error trace is returned.

The research project applied model checking to deriving test suite from real-time system design models for Programmable Logic Controllers (PLCs), so as to improve the quality of the model-based testing. After completing the projects in the *RTES* course, students can formally verify Simulink models and generate test suites from the models to test the PLC program. Three open source software tools have been employed in this course for students to conduct projects: two model checkers (Bounded Model Checker for C – CBMC [16] and model checker for real-time systems - Uppaal [17])

and the code generator Gene-Auto [18]. The projects shown in Table 4 build on each other and follow the order of the conducted research. Students are required to complete the model-based verification via a model checker in the first two projects. In the last two projects, students should finish the mutation-based test-case generation for model-based testing. To be specific, firstly, students should complete the project of translating the given Simulink diagram to the C code for the verification purpose by the use of the Gene-Auto tool. A project report is required for students to trace the mapping relationship between the blocks in the Simulink diagram and the translated functions in C. Then, students are required to formulate safety properties under verification as assertions in C and use the tool CBMC to check if the properties hold in the diagram. Project 3 is designed for students to make four logic mutations in the original design diagram, for example, the "AND" logic operator block is mutated to the "OR", "NOR", "NAND", or "XOR" block to emulate the common faults programmers may make. Project 4 is to apply the counterexample returned from the model checker to generate test cases for the model-based testing.

Table 4. Projects for the RTES course

Project Description	**Learning outcomes**
1. Use an existing code generation tool to translate the Simulink diagrams to C code.	- Investigate the automatic code generation mechanism and analyze the mapping relationship between Simulink design and implementation.
2. Use a bounded model checker (i.e. CBMC) to formally verify safety properties of the C Code.	- Formulate the safety properties and apply a model checker to ensure the correctness of the actual design or C code.
3. Insert the mutations into the design to create a set of mutants and build miter models.	- Apply a fault-based testing technique for the mutation testing.
4. Apply a bounded model checker to miter models for test-case generation from the returned counterexamples.	- Apply the model checking technique to generate the test cases via mutation testing.

V. Evaluations

The course evaluation surveys have been conducted to assess the teaching effectiveness of the MBD techniques in two existing courses. In Table 5, the two percentage numbers in each cell represent the results from the *PT* and *RTES* course in the form of *PT/RTES*. Most students from the *PT* course comment that the MBD method gave them a new view of designing and programming the MCUs and increased their confidence in the embedded programming. Students in the RTES course reported that the subject of the model-based V&V was mostly very helpful to study the advanced verification techniques including model checking and mutation-based testing.

Table 5. Summary of students' evaluation

Survey Questions	**Excellent [%]**	**Good [%]**	**Fair [%]**	**Poor [%]**
Understanding and applying the MBD	80/70	20/30	0/0	0/0
Instructor's help via examples and projects	90/60	10/40	0/0	0/0

VI. Conclusion and Future Work

This paper presents our experiences of applying the PBL approach to teaching engineering and technology students the MBD in two existing courses. The course materials, especially course projects, are derived from two on-going research projects. Through projects, students learn the research skills besides the advanced knowledge of the MBD. In the future, we will apply MBD and Simulink to program new hardware platforms like Raspberry Pi and encourage students to conduct the research on the modern model-based V&V techniques.

References

[1] V. Gonzalez and E. Freudenthal, "SDIm an introductory programming course with an ECE context," *Frontiers in Education Conference (FIE), 2010 IEEE*, Washington, DC, pp. S2F-1-S2F-2.

[2] A. Gomes and A. Mendes, "A teacher's view about introductory programming teaching and learning: Difficulties, strategies and motivations," *Frontiers in Education Conference (FIE), 2014 IEEE*, Madrid, pp. 592-599.

[3] EU CESAR project (Cost-Efficient Methods and Processors for Safety Relevant Embedded Systems) http://www.cesarproject.eu/

[4] EU MOGENTES project (Model-based Generation of Tests for Dependable Embedded Systems) http://www.mogentes.eu/

[5] C. Stone and S. Strong, "Implementing & evaluating Undergraduate Research in renewable energy at Colorado School of Mines," *Frontiers in Education Conference (FIE), 2012 IEEE*, Seattle, WA, pp. 197-202.

[6] M. R. Anderson-Rowland, A. A. Rodriguez and A. E. Grierson, "Helping undergraduate engineering students discover their interests (and reduce their fear of research)," *Frontiers in Education Conference (FIE), 2015 IEEE*, El Paso, TX, pp. 1537 - 1544.

[7] N.He. "Incorporating On-going Verification & Validation Research to a Reliable Real-Time Embedded Systems Course". In proc. of the ASEE North Midwest Sectional Conference, Fargo, 2013.

[8] E. Clarke, O. Grumberg, D. Peled, *Model Checking* (MIT Press, 1999).

[9] R. C. Bryce, A. Cooley, A. Hansen and N. Hayrapetyan, "A one year empirical study of student programming bugs," *Frontiers in Education Conference (FIE), 2010 IEEE*, Washington, DC, pp. F1G-1-F1G-7.

[10] B.J. Smith. "*Using a Visual Programming Language to Bridge the Cognitive Gap between a Novice's Mental Model and Program Code*." Ph.D. Dissertation. Univ. of Alabama in Huntsville, AL, USA, 2011.

[11] O. L. Oliveira, A. M. Monteiro and N. T. Roman, "Can natural language be utilized in the learning of programming fundamentals?," *Frontiers in Education Conference, 2013 IEEE*, Oklahoma OK, pp. 1851-1856.

[12] R. A. P. Oliveira, L. B. R. Oliveira, B. B. P. Cafeo and V. H. S. Durelli, "Evaluation and assessment of effects on exploring mutation testing in programming courses," *Frontiers in Education Conference (FIE), 2015 IEEE*, El Paso, TX, pp. 179-187.

[13] O. Ochoa and S. Salamah, "An approach to enhance students' competency in software verification techniques," *Frontiers in Education Conference (FIE), 2015 IEEE*, TX, pp. 170-178.

[14] P. Jamieson and J. Herdtner, "More missing the Boat — Arduino, Raspberry Pi, and small prototyping boards and engineering education needs them," *Frontiers in Education Conference (FIE), 2015 IEEE*, El Paso, TX, pp. 1442-14447.

[15] J. W. Thomas. "A review of research on project-based learning". http://www.bie.org/index.php/site/RE/pbl_research/29, 2000.

[16] E.M. Clarke, D. Kroening and F. Lerda, "A tool for checking ANSI-C programs", in Proceedings of *TACAS*, pp168-176, Springer, 2004.

[17] K. Larsen, P. Pettersson and W. Yi. "Uppaal in a nutshell". Intl. Journal on Software Tool sfor Technology Transfer (STTT), pp134-152, 1997.

[18] A. E. Rugina and J. C. Dalbin. "Experiences with the Gene-Auto Code Generator in the Aerospace Industry". *Proceedings of Intl. Conf. on Embedded Real-Time Software and Systems ERTS2*, 2010.