Teaching Portfolio submitted by Abhik Roychoudhury

(part of Supplementary Dossier)

Email: abhik@comp.nus.edu.sg
http://www.comp.nus.edu.sg/~abhik

CONTENTS

I. Contribution to Module Development by Abhik Roychoudhury

Email: abhik@comp.nus.edu.sg
http://www.comp.nus.edu.sg/~abhik

During my employment at NUS since 2001, I have contributed to the development of several modules. Some of these modules were developed from scratch, while others were a very substantial revamp from the past content. In the following, I first highlight those modules whose development was done by me from scratch. In addition, I have also included at least one module whose content was substantially re-vamped by me.

A. Modules whose contents were developed by me from scratch, in recent years

CS 4271 Critical Systems and their Verification [Module folder included]

This is a module that was developed by me, from scratch, not once, but twice. Around 2002, I offered this module for the first time, and designed the course content from scratch. In its first version, the module focused on automated verification methods, specifically model checking. This offering of the module covered topics like

Modeling of behaviors via transition systems, Temporal Logics, Explicit-state Model checking, Binary Decision Diagrams, Symbolic Model Checking and Fairness Issues in Model Checking.

Thus, the earlier offering of the module was more of a theoretical nature. Subsequently around 2007-08, I broad-based the module contents to cover both formal verification and less formal validation methods such as testing. Furthermore, given the focus of the module on computer engineering students – I broad-based the module contents to cover validation of non-functional properties, apart from functionality validation. In the process of this module revision, I authored a text-book "Embedded Systems and Software Validation" which is described later in this portfolio. The revised contents of the module cover

Role of system modeling, Modeling notations, Model-based testing, Model checking, Familiarization with selected model checkers, Performance validation at software level, Performance Validation at System level (Scheduling), Functionality validation at software level – testing and debugging.

More details about the module content development also appear in the next section of the port-folio, where I discuss the experience in authoring the text-book. A web-site of an earlier offering of the module can also be accessed from http://www.comp.nus.edu.sg/~abhik/CS4271/lesson-plan.html

CS6880 Advanced Topics in Software Engineering

I offered a new module Advanced Topics in Software Engineering in 2011-12. This module involved paper presentations (with peer review by other students), projects and discussions, apart from lecturing. A broad based offering covering advanced theoretical concepts (such as link between logics and requirements) and the advanced state-of-the-practice (such as the testing methods used in industry) was worked out. In terms of the contents of the module CS 6880, I offered a mix of advanced theoretical concepts, as well as advanced concepts in the state-of-the-practice in software engineering. In terms of advanced theoretical concepts, we studied the underlying foundations of temporal logics and their linkage in terms of representing behavior as captured in software requirements. In terms of advanced practical topics, I covered testing methods studied in the industry for safety-critical software, such as MC/DC testing. Apart from conventional lecturing, this module also involved independent projects and paper presentations. The students in this module did projects in cutting edge software engineering issues such as: Software out-sourcing, software-as-a-service (SaaS) and software taint analysis. The paper presentations were done by the students and also involved peer-review from other students.

CS5219 Automated Software Validation

In this module, the students are exposed to a wide variety of methods for formal verification of software. These methods are particularly important for software deployed in safety-critical settings. The techniques employed range from search based techniques such as model checking to more powerful automated theorem proving methods. The students undertake a substantial project where they take informal English language requirements, develop a formal model and employ state-of-the-art model checking tools to perform formal verification of properties of the system model. More advanced students are also encouraged to perform automated code generation from the formally verified model.

In various offerings of the module, I have offered various engaging course projects. One can refer to http://www.comp.nus.edu.sg/~abhik/5219/lesson-plan.html to get a feel for the structure of the module. In this offering, we gave one of the challenge problems in software verification to the students in the module. This involved modeling, verifying and reliably developing the software for a pacemaker http://sqrl.mcmaster.ca/pacemaker.htm based on real-life requirements specification of a pacemaker made available by a company. This gives the students valuable exposure to the challenges in developing correct software from informal requirements.

CS4218 Software Testing and Debugging

I am currently developing an undergraduate module on Software testing and debugging, which will be offered in Jan 2014. This proposed module will be a primary module in the Software Engineering focus area in the CS department. This is also the first time that a module on the fundamentally important area of Software Testing will be taught at NUS. At this point, I have finished a survey of existing courses and teaching methods in software testing through my personal contacts among software testing and software engineering researchers. The course will be fully project based rather than having a collection of assignments. During the duration of the course, every attempt will be made to simulate industrial setups such as one project group fixing bugs in another project group's code. My discussions with software engineering researchers at industrial labs, such as Microsoft Research, has confirmed that bringing in such industrial practices into curriculum makes the students much more suited for the work-place. Finally, I also plan to include discussion of cutting-edge testing research in the course. At this point, I am actively involved in research on testing and debugging, and the students usually appreciate getting a feel of the most cutting edge technologies in the field.

B. Modules whose contents were substantially revamped by me, in recent years

CS3211 Parallel and Concurrent Programming [Module folder included]

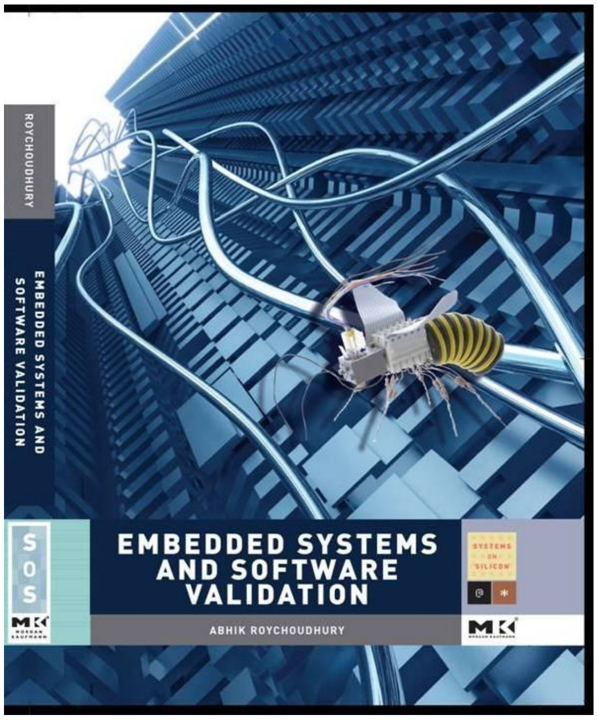
A concurrent system consists of set of processes which execute simultaneously and may collaborate by communicating and synchronizing with each other. This leads to subtle program behaviors which the students need to learn how to grasp, via this module. This is a module offered to third year undergraduate students; some of the students attending the module might also be in their second year of study. The module provides a second exposure to the powerful concept of concurrency – with the students having received a very initial exposure earlier in a module on operating systems. The students learn concurrent programming primitives such as locks, barriers and also study the realization of such primitives in a mainstream multi-threaded programming language such as Java. In the later part of the module, the students appreciate the nuances separating concurrency and parallelism, and take part in hands-on parallel programming on top of a popular programming language.

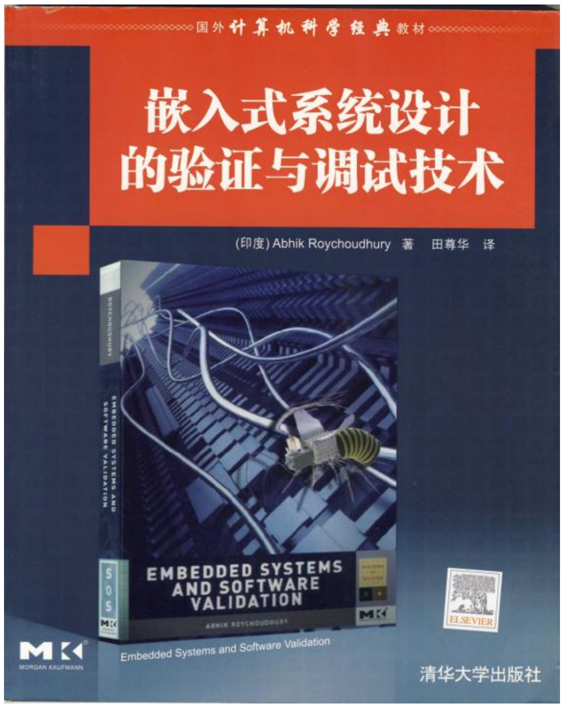
The earlier offering of the module focused less on programming and more on formal concurrency modeling as well formal verification of concurrent systems. In my re-vamp of the module, I made the module contents much more hands-on, and put the focus on concurrent and parallel *programming*! Most of the lecture/tutorial/assignment contents were focused on multi-threaded programming using Java, and parallel programming using MPI. Each of the major notions such as threads, locks, monitors, message passing – were first discussed conceptually and then followed up immediately thereafter with hands-on-programming.

The module folder of CS 3211 is being made available with this portfolio. A web-site of the module is also available from http://www.comp.nus.edu.sg/~abhik/CS3211/2013/index.html

II. Authoring of text-book by Abhik Roychoudhury

Email: abhik@comp.nus.edu.sg
http://www.comp.nus.edu.sg/~abhik


ROYCHOUDHURY
EMBEDDED SYSTEMS AND SOFTWARE VALIDATION
SOS
EMBEDDED SYSTEMS AND SOFTWARE VALIDATION
ABHIK ROYCHOUDHURY
SYSTEMS ON SILICON
MK MORGAN KAUFMANN


国外计算机科学经典教材
嵌入式系统设计的验证与调试技术
(印度) Abhik Roychoudhury 著 田尊华 译
EMBEDDED SYSTEMS AND SOFTWARE VALIDATION
Embedded Systems and Software Validation
MK MORGAN KAUFMANN
ELSEVIER
清华大学出版社

Book-cover Book-cover of the Chinese translation

While teaching the module CS4271 Critical Systems and their Verification for several years – I became acutely aware of the need for a text-book in this area. There were indeed several books on formal verification or model checking – which I used to consult in my earlier offerings of the module. However, these books completely lacked the system design perspective and described the material on model checking from a theoretical perspective. On the other hand, textbooks on embedded systems did not contain a rigorous design and verification perspective. Around 2007, I was approached by Morgan Kaufmann (Elsevier) to write a textbook on this topic under their prestigious Systems-on-Silicon series. I undertook the effort, and the textbook has subsequently been adopted in various universities in Europe, USA and Asia. The book was published in January 2009

A Chinese translation of the book was initiated by Tsinghua University Press in 2011. The Chinese translation has been adopted for teaching in Chinese universities.

Excerpts from the book were featured as “Editors Top Picks” in EE times, a popular industry magazine with embedded developers. This appeared in the edition on 6th August 2012.

A. From the book cover

Modern embedded systems are a part of every modern electronic device, ranging from toys to traffic lights to nuclear power plant controllers. These systems help run factories, manage weapon systems and enable the worldwide flow of information, products and people. Unlike other computer systems such as those that operate personal computers, embedded systems must typically run error-free for years or even decades with little or no opportunity to reboot the system or fix problems. Such systems typically consist of a heterogeneous collection of processors, specialized memory subsystems and partially programmable or fixed-function components. This heterogeneity, coupled with issues such as hardware/software partitioning, mapping and scheduling leads to a large number of design possibilities, making both functionality and performance validation of such systems a difficult problem and an imperative issue.

Roychoudhury guides readers through a host of debugging and verification methods critical to providing reliable software and systems applications. Readers will find practical information and guidance including:

- Complete coverage of the major abstraction levels of embedded systems design, starting from software analysis and micro-architectural modeling, to modeling of resource sharing and communication at the system level;
- Integrates formal techniques for validation for hardware/software with debugging methods for embedded systems;
- Includes practical case studies to answer the questions: does a design meet its requirements, if not, then which parts of the system are responsible for the violation, and once they are identified, then how should the design be suitably modified?

B. Preface of the book – explaining its differentiation with other books

This book attempts cover the issues in validation of embedded software and systems. There are lot of books on “embedded software and systems" as a web search with the appropriate search terms will reveal. So, why this book?

There are several ways to answer the question. The first, most direct, answer is that the current books mostly deal with the programming and/or co-design of embedded systems. Validation is often discussed almost as an after-thought. In this book, we treat validation as a first class citizen in the design process, weaving it into the design process itself.

The focus of our book is on validation, but from a embedded software and systems perspective. The methods we have covered (testing/model-checking) can also be covered from a completely general perspective, focusing only on the techniques, rather than how they fit into the system design process. But we have not done so. Even though the focus of the book is on validation methods, we clearly show how it fits into system design. As an example, we present and discuss the model checking method twice in two different ways - once at the level of system model (Chapter 2) and again at the level of system implementation (Chapter 5). Finally, being rooted in embedded software and systems, the focus of our book is not restricted to functionality validation. We have covered at least two other aspects - debugging of performance and communication behavior. As a result, this book contains analysis methods which are rarely found in a single book - testing (informal validation), model checking (formal validation), worst-case execution time analysis (static analysis for program performance), schedulability analysis (system level performance analysis) and so on - all blended under one cover, with the goal of reliable embedded system design.

As for the chapters of the book, Chapter 1 gives a general introduction to the issues in embedded system validation. Differences between functionality and performance validation are discussed at a general level.

Chapter 2 discusses model-level validation. It starts with a generic discussion on system structure and behavior and zooms into behavioral modeling notations such as Finite-state machines (FSMs) and Message Sequence Charts (MSCs). Simulation, testing and formal verification of these models are discussed. We discuss model-based testing, where test cases generated from the model are tried out on the system implementation. We also discuss property verification, and the well-known model checking method. The chapter wraps with a nice hands-on discussion on practical validation tools such as SPIN and SMV. Thus, this chapter corresponds to model-level debugging.

Chapter 3 discusses the issues in resolving communication incompatibilities between embedded system components. We discuss different strategies for resolving such incompatibilities, such as endowing the components with appropriate interfaces, and/or constructing a centralized communication protocol converter. Thus, this chapter corresponds to communication debugging.

Chapter 4 discusses system level performance validation. We start with software timing analysis, in particular Worst-case Execution Time (WCET) analysis. This is followed by the estimation of time spent due to different interferences in a program execution from the external environment, or due to other executing programs on same/different processing elements. Suitable analysis methods to estimate the time due to such interferences are discussed. We then discuss mechanisms to combat execution time unpredictability via system level support. In particular, we discuss compiler controlled memories or scratchpad memories. The chapter concludes with a discussion on time predictability issues in emerging applications. Thus, this chapter corresponds to performance debugging.

Chapter 5 discusses functionality debugging of embedded software. We discuss both formal and informal approaches, with almost equal emphasis on testing and formal verification. The first half of the chapter involves validation methods built on testing or dynamic analysis. The second half of the chapter concentrates on formal verification, in particular, software model checking. The chapter concludes with a discussion on combining formal verification with testing. Thus, this chapter corresponds to software debugging.

Apart from some debugging/validation methods being common to Chapters 2 and 5, the readers may try to read the chapters independently. A senior undergraduate or graduate course on this topic may however read the chapters in sequence, that is, chapters 2, 3, 4, 5.

C. Invitation e-mail from Morgan Kaufmann (Elsevier) for writing the book

Following is the e-mail invitation I received from Morgan Kaufmann for writing a book on debugging and validation of embedded systems.

-------- Original Message --------

Subject: Debugging Embedded Systems...the BOOK?

Date: Wed, 17 Jan 2007 10:17:27 -0600

From: Glaser, Chuck (ELS-BUR) <C.Glaser@elsevier.com>

To:<abhik@comp.nus.edu.sg>

Dear Professor Roychoudhury:
I have taken the liberty of writing you today in my capacity as Senior Editor with Elsevier. I work closely with Wayne Wolf on the publication of our Morgan Kaufmann Series in Systems on Silicon. I noticed with keen interest that you gave one of the tutorials at VLSI 2007 on "Performance Debugging of Complex Embedded Systems" and I wonder if you have given any thought to working on a book-length treatment of this topic, either as author or as editor, with various, contributing authors?

If you would be interested in working on a Debugging Embedded Systems book, or if there are other book projects growing out of your current research, I would be very pleased to discuss the details with you.

Thank you in advance for your consideration.

Sincerely,
Chuck Glaser

PS: FYI, I have attached two files describing the series and listing titles…. Also, fyi, we pay our authors competitive royalties, we price our books to market (the most expensive in the series is $79.95), and we produce our books to the highest possible standards. I hope we can talk!

****
Charles B. Glaser
Senior Acquisitions Editor
Elsevier

D. Comments from instructors who adopted the book in their modules

Finally, following are some sample comments from instructors who have adopted my book in their teaching (as communicated to me via e-mail).

-----Original Message-----
From: Abhik Roychoudhury [mailto:abhik@comp.nus.edu.sg]
Sent: Tuesday, September 15, 2009 10:41 AM
To: J. Park
Subject: Re: Materials on your book

J. Park wrote:

Dear Prof. Roychoudhury:

I've chosen your book as a textbook of my lecture on the developing
an embedded software.
I think this book is good for the concept and practical issues on
embedded software verification.
I'm just wondering whether you have a kind of teaching materials
such as powerpoint file or figures used in your book?

If you could provide any kind of teaching material, it would be
greatly helpful for me to prepare my lecture.
Thank you very much for such a great book.
Regards,

Jaehyun Park
Professor
Inha University, Korea

-------- Original Message --------

Subject: About your book

Date: Wed, 29 Jun 2011 09:11:57 +0530

From: Pallab Dasgupta <pallab@cse.iitkgp.ernet.in>

To:<abhik@comp.nus.edu.sg>

Dear Abhik,

This is to congratulate you for your book "Embedded Systems and Software Validation", which is a wonderful contribution in an area where very few texts are available. You work is a great contribution to academia in this area.

I shall be teaching a course component on Embedded Systems and Software Validation in the forthcoming Autumn semester, for which I intend to use your book as the main reference. I got the link to this book from Dr Ansuman Banerjee and have been able to procure this book from our bookstore.

Ansuman told me that you share your powerpoint slides for this book -- may I get these?

Kind regards,

Pallab

==
Dr. Pallab Dasgupta,
Professor, Dept. of Computer Sc & Engg,
Indian Institute of Technology Kharagpur, INDIA 721302.

Phone: +91-3222-283470 (Off) 283471 (Res)
Fax: +91-3222-278985
Email: pallab@cse.iitkgp.ernet.in
Web: http://www.facweb.iitkgp.ernet.in/~pallab
==

III. Future Plans in Teaching

Email: abhik@comp.nus.edu.sg
http://www.comp.nus.edu.sg/~abhik

"*Those who know, do. Those that understand, teach.*" *- Aristotle.*

Teaching is well understood to be an engaging and enriching activity. At the same time, teaching and learning being closely inter-twined, teaching remains a crucial source of reflection. In writing about my future plans on teaching, it allows me to reflect not only my own teaching, but also on ongoing / future developments in computing which are likely to have a profound impact on teaching of future generations.

Massive open online courses (MooC) have attracted a lot of attention in the recent years. These online courses allow for large scale and interactive participation facilitated via the internet. MooCs are undoubtedly popular among students who would not have access to a quality university education. However, for students who are already enrolled in a high-quality university like NUS, what differentiation does the NUS teaching provide over online MooCs? I believe it is important to understand some of the key advantages of MooCs, if we want to deliver clear value addition to our classroom teaching. The major advantages of MooCs include – easy accessibility of educational material, ability of the students to repeatedly go over the educational material, and learning by exercises and reflection as opposed to content delivery. The first two advantages can be easily provided in a high-quality university environment – the university has lecturers who produce educational material and make it accessible to students, and lectures can be recorded for students to go over the material. However, the third advantage of MooCs may not always be straightforward to reproduce in a classroom teaching environment. Often times in a 13 week semester, we spend the lecture time discussing the module content, often through examples and discussions. In the tutorials, the student do get exposed some exercises, which are previously circulated. However, there is less contact hours on reflections – where the students and lecturer can get together to not only solve problems, but also critically review the fundamental concepts. I have tried this practice in my latest offering of CS3211 Parallel and Concurrent Programming. Over and above the lecture and tutorial sessions – we also had additional reflection sessions on Saturdays. In these reflection sessions, either (a) we did a critical review of the basic concurrency constructs, such as locks, message passing etc, and discussed how they are implemented in a real system, or (b) we solved problems which were given on-the-fly and were not previously circulated. This worked reasonably well in the first half of the semester, however in the second half of the semester – the students gradually got busy with other modules towards the end of the semester. Looking forward into the future, I will seek to integrate such additional reflection sessions more and more into my teaching.

I am also working on developing teaching material which integrates more of the industry practice in software development. In my upcoming module CS 4218 Software Testing and Debugging (to be offered in January 2014) – I am simulating industrial practices where there are well-defined roles for "tester" and "developer". Thus, when students work in a company, they may need to test, debug and understand code written by others. To prepare the students for such ground realities, I have prepared the module to have a course project with two parts. In the first half of the semesters, the students develop code in teams, and then in the second half of the semester they hand over their code to another team. The other team then tests/debugs the code handed over to them, and generates quality-control report. The students are graded for adopting best practices in development, testing and debugging. They are also prepared for the situation in companies where most of the development consists of changes to a large code-base, as opposed to writing code from scratch. In future, I plan to work more and more towards integrating such industry practices, in the design of modules. This helps the student better understand ground realities.

IV. E-mails from students

Email: abhik@comp.nus.edu.sg

http://www.comp.nus.edu.sg/~abhik

Here are some emails from students in my modules. These emails were sent after module is completed.

-------- Original Message --------

Subject: Hello Sir

Date: Mon, 31 May 2010 11:04:22 +0800

From: Sridhar, Mihir (Equity Technology) <mihir.sridhar@baml.com>

To:abhik@comp.nus.edu.sg

CC:csmihir@nus.edu.sg

Hi Sir,

Mihir here… the results came out earlier today, and I was able to get a SAP of 5/5.

Wanted to thank you for all your assistance during the course of both the modules. I scored an 'A' for CS3211, and an 'A+' for CS4271. Though my mid-term results weren't too great, I think your encouragement made me work much harder during the 2nd half of the courses.

Hoping to continue to stay in touch with you. I am currently working with Bank Of America Merrill Lynch as a Summer Associate for the summer. I will be coming back to school for my final semester in August, and I graduate in December this year.

Thanks once again. I am hoping I can perhaps meet up with you sometime. Would like to seek some inputs from you about my future plans and such.

Regards,

PS: Please reply-all, as I cant access my work emails from home…

Warm Regards,
Mihir **SRIDHAR** *(Mr.) | Summer Associate*
Equity Link Technology (ELT)
Global Markets and Research Technology (GMRT)
Ph: +65.6591.1601 | Email: mihir.sridhar@baml.com

-------- Original Message --------
Subject:CS3211
Date: Fri, 10 May 2013 12:12:00 +0800
From: Ngik Kok Wai Vincent <vincentngik@nus.edu.sg>
To:abhik@comp.nus.edu.sg <abhik@comp.nus.edu.sg>

Dear Sir,

I would like to thank you for the interesting and wonderful experience I had over the course of this module. I have certainly learn a lot of things from the module. The first time I have to deal with parallel and concurrent programming was when I was at my attachment at Microsoft during my internship last summer. That experience has sparked my interest and make me inclined to take this module. I am glad that I chose to take this as I had a better understanding of parallel and concurrent programming. Once again, I would like to express my gratitude for the enriching module.

Regards,
Vincent Ngik

Date: Tue, 4 May 2010 16:12:28 +0800
From: Shaun Chong <shihwai@nus.edu.sg>
To:abhik@comp.nus.edu.sg

Thank you Prof,

From the bottom my my heart, you are truly a great professor helping us understand this cryptic and notoriously difficult module with ease.

Shaun