I invited Mike Huth, Imperial College London to be the guest editor for this column. What started as a simple idea evolved into a complete paper. I encourage you to read Mike’s insightful article “Mathematics for the exploration of requirements” in this InRoads issue. Below is a little background to ease you into an area that is becoming relevant and important in software systems development – modeling.

The noun model means “a miniature representation of something; or a pattern of something to be made”. In engineering there are both physical models and mathematical models. Indeed, mathematics is a key tool for modeling in most disciplines. Evidence for this in undergraduate education comes from the MAA CUPM Curriculum Foundations Project (http://www.maa.org/cupm/crafty/) in which only one topic, ‘modeling’ was found to be significant for all client disciplines - engineering, economics, computer science, physics, chemistry, biology, business, manufacturing, statistics, mathematics, etc.

Engineers often construct both physical and mathematical models, and compare these to help validate the models. Modeling building and tweaking are fundamental for validating the desired features and behavior, before design. Software is abstract in nature so physical models are usually not constructed. The models our students build are usually intuitive and mental. They often don’t have the mathematical and/or software tools required for model building, testing, tweaking, experimenting and validation prior to design. Indeed, the software system they develop is frequently the “first executable model” available for experimenting, testing, tweaking, etc. This “model” is overly prescribed, detailed and unwieldy.

A key role of mathematics in CS and SE education is modeling. The simple task of understanding an informal requirement such as “the compiler responds with an error message when a variable is not declared” is modeling – making it precise is mathematical modeling. Mathematics is an important tool for constructing a working compiler where it is an integral part of lexical analyzers, parsers, code generators, etc.

Mathematically based system modeling tools for developing executable and verifiable models are beginning to evolve, and I predict will become standard practice for the development of software based systems which exhibit a high degree of reliability. Two articles addressing these issues are in the Frontiers in Education conference series: “Logic in Computer Science: tool-based modeling and reasoning about systems” by Mike Huth, FIE 2001, (http://fie.engrng.pitt.edu/fie2001/ Session T1C) and “The Role of Modeling in Software Engineering Education” by yours truly, FIE 2003 (http://fie.engrng.pitt.edu/fie2003/ Session S1C).
With the waterfall process of software engineering, requirements, which are one way of capturing a model of a system, need to be completely specified. However, for many software systems being developed or maintained, it is difficult to completely specify the requirements. Software evolution is often incremental, requiring extensive feedback. Accordingly, a tool for specifying, experimenting with, analyzing, testing, etc. an incompletely specified model of the system requirements would be nice to have. This is the thrust of Mike Huth’s article “Mathematics for the exploration of requirements” and I encourage you to read it.

More on the important relationships between mathematics and modeling in the next Math CountS column.

**Future events of interest:**

1. **Nifty Applications in Discrete Mathematics** a Mathematical Association of America Professional Enhancement Program workshop organized by Bill Marion to be held June 7-11, 2004 at Valparaiso University for mathematics and computer science faculty teaching introductory discrete mathematics courses for computer science and software engineering majors. See [http://www.maa.org/prep/](http://www.maa.org/prep/) for details and registration information.

**Summary of recent events and activities related to mathematics in CS and SE education:**

1. The **SIGCSE Committee on the Implementation of a Discrete Mathematics Course** was approved by the SIGCSE board July 2003. Doug Baldwin and Bill Marion are the co-facilitators for this committee. The charge of the committee can be found at [http://www.sigcse.org/topics/committees.shtml](http://www.sigcse.org/topics/committees.shtml). To join the committees discussion listserv by sending the message ‘SUBSCRIBE SIGCSE-MATH-COMM your name’ to LISTSERV@ACM.ORG. Here is a committee report from Bill Marion.

The charge to the committee is to develop a small number of models for a one-semester discrete mathematics course that incorporates the CC 2001 core discrete topics. A committee listserv was set up and over sixty mathematics and computer science faculty responded to an invitation to join the committee. After some initial conversation among the members the co-facilitators, we decided that it would be a good idea (so as not to reinvent the wheel) to develop a brief survey to find out what type of discrete math courses were already being taught.

During in the fall of 2003 the survey was sent to a number of math and CS listservers. We received over one hundred responses, about 80% of which came from colleges and universities where a one-semester course is offered. Preliminary analysis, grouped by type of institution, reveals that about 60% of the colleges and universities with a one-semester course have separate computer science departments (as opposed to joint math/cs departments) and about an equal number of math faculty teaching the course as computer science faculty. In addition, about half of the responses indicated that the discrete math course is being taught in the first year.

Prerequisites for the course, irrespective of what year it is taught, run the gamut from College Algebra to Calculus II--about 20% of these courses also have a CS I-type course.
as a prerequisite. About 60% of the one-semester respondents to the survey attached a course syllabus or a web address where one can be found. The next step is to review those syllabi to tease out various models.

2. **Responses to Mathematics in CS Education** (September 2003 issue of the Communications of the ACM) – The letters in response to the articles in this special issue of the CACM were mixed and very interesting. I exchanged email with several of the respondents to further clarify their positions, and perhaps mine. What I found most interesting is that a few of the respondents got the impression that the issue was advocating formal methods and made the point that such methods are not yet mature or proven.

My view is that the issue was about using mathematics to reason precisely about software, something CS and SE students seem to have a lot of difficulty doing, I think. Indeed, I feel the term “formal methods” should be abolished. CS and SE curricula should integrate math based logical reasoning throughout all courses as is done in engineering curricula.

3. **17th Conference on Software Engineering Education and Training (CSEE&T 2004)** – March 1-3 in Norfolk, VA, USA. (http://www.cs.virginia.edu/~cseet04/) There is a session devoted to formalism that includes 3 papers with the following titles: “Software Engineering Education Needs Adequate Modeling Tools”, “Integrating Design Formalisms in Software Engineering Education”, and “A Case Study Involving the Use of Z to Aid Requirements Specification in the Software Engineering Course”.

   
a. A workshop “Nifty Examples in Discrete Mathematics” by Peter Henderson and Bill Marion.
c. “Mathematical Reasoning in Computer Science” a Birds of a Feather session organized by Doug Baldwin and Peter Henderson.
d. “Teaching Growth of Functions Using Equivalence Classes An Alternative to Big O Notation” a paper by Constantine Roussos.
e. “We Claim this Class for Computer Science. A Non-Mathematician’s Discrete Structures Course” by Adrienne Decker and Phil Ventura.

5. I hope everyone has had a chance to see the excellent movie **“To Dream Tomorrow”** about the life and contributions of Ada Lovelace. Ada is considered by many to be the first computer programmer. I encourage your institutions library, or your college or department, to purchase this 52 minute movie. Well worth the $100 - see http://www.mith.umd.edu/flare/lovelace/ for details. Show it every year to your students.

Previous Math CountS columns are available at http://blue.butler.edu/~phenders/InRoads/
This document was created with Win2PDF available at http://www.daneprairie.com. The unregistered version of Win2PDF is for evaluation or non-commercial use only.