

# A Five Year Perspective on Software Engineering Graduate Programs at George Mason University

Paul Ammann, Hassan Gomaa, Jeff Offutt, David Rine, and Bo Sanden  
Department of Information and Software Systems Engineering  
George Mason University  
Fairfax, VA 22030-4444  
phone: 703-993-1640  
email: hgomaa@isse.gmu.edu

May 1993

## Abstract

This paper describes the experience obtained at George Mason University while developing a Master of Science program in software engineering. To date, the program has graduated over 45 students, with a current production rate of 10 to 15 a year. The paper also describes experience with a certificate program in software engineering, which is a software engineering specialization taken by Masters students in related disciplines, and the software engineering specialization within the PhD program in Information Technology. We discuss our courses, students, the successes that we have had, and the problems that we have faced.

## 1 INTRODUCTION

George Mason University (GMU), which is situated in suburban Washington D.C., has more than 21,000 students, including 5,500 master's students, 1,100 doctoral students and 2,000 post-baccalaureate extended studies students. Of a total of 1,400 faculty, 643 are full-time. GMU offers 35 master's programs, 12 doctoral programs and 8 certificate programs.

The School of Information Technology and Engineering (SITE), established in 1985, houses the departments of Computer Science (CS), Information and Software Systems Engineering (ISSE), Systems Engineering (SE), Electrical and Computer Engineering (ECE), Operations Research and Engineering (ORE), and Applied and Engineering Statistics (AES). The emphasis is on graduate education; whereas four BS programs produce a total of 220 graduates yearly, primarily CS and ECE, seven MS programs – CS, ECE, Information Systems (IS), Operations Research, Software Systems Engineering (SWSE), Statistical Science, and Systems Engineering (SE) – graduate 250 students annually. A school-wide doctoral program in Information Technology and Engineering graduated 13 PhDs in academic year 1992, and 20 in 1993.

Part of GMU's unique mission is to serve the Metropolitan Washington and Northern Virginia area. The Master of Science program in Software Systems Engineering (MS-SWSE) was established at George Mason University in an attempt to respond to the needs of high technology industries in the area. The discipline of software engineering is concerned with technical and managerial issues related to developing and modifying complex, large scale computer software systems. The purpose of the program is to train students in the theory and practice of software engineering.

One important aspect of GMU's location, impacting our program in many and diverse ways, is that the majority of our students are part-time students who work for high technology companies, research laboratories, consulting firms, and federal government agencies located in the Metropolitan Washington area. The most obvious result of this is that all of our courses meet once a week, in the late afternoon or early evening. Additionally, many of our students have their tuition paid by their employers.

## **2 THE SOFTWARE SYSTEMS ENGINEERING MASTER'S PROGRAM**

The Master of Science program in Software Systems Engineering (MS-SWSE) was established at George Mason University in the fall of 1989, a year after the introduction of the Certificate program in Software Systems Engineering. Planning for the MS and certificate programs started in the fall of 1987 when three of the faculty of the Wang Institute of Graduate Studies moved to George Mason after the Wang Institute closed down in the summer of 1987. A three year \$2.4 million grant from the Virginia Center of Innovative Technology was instrumental in providing the funding basis for establishing a software engineering group at George Mason University.

The masters and certificate programs are housed in the Department of Information and Software Systems Engineering (ISSE) and supported by the ISSE and CS departments. ISSE also offers a related MS program in Information Systems. Many of the software engineering courses are cross-listed with the Computer Science Department, which often provides faculty to help teach the courses.

To receive the Masters degree in Software Systems Engineering, students must complete 30 semester hours of graduate work. Students have a choice of a professional track of six core courses and four electives, or a research track of six core courses, two electives, and a six hour Masters thesis. The core courses are:

- Software Construction
- Software Requirements and Prototyping

- Software Design
- Formal Methods and Models
- Software Project Management
- Software Project Lab

We are currently considering expanding the core to include Software Testing and Quality Assurance.

Students may choose either software engineering electives or electives from related disciplines. Software engineering electives include Software Engineering Economics, Object-Oriented Software Development, User Interface Design and Development, Software Testing and Quality Assurance, Advanced Software Requirements, Advanced Software Design Methods, Special Topics in Software Engineering, and Directed Readings in Software Engineering. A wide range of electives is also available in the areas of computer science, electrical and computer engineering, systems engineering, etc.

GMU's current program represents an evolution from the Wang Institute Master of Software Engineering program [Ard87, Fai88a, ADF<sup>+</sup>91, Fai88b]. The Formal Methods and Models, Software Project Management, and Software Project Lab all existed at the Wang Institute. Software Construction replaces the Wang Institute course on Programming Methodology.

The Wang Institute course in Software Engineering Methods, which emphasized requirements and design, evolved into a Software Requirements, Prototyping and Design course at GMU. However, it was soon found that there was insufficient time to do justice to both requirements and design in one course, and so the course was split into the two courses now being offered: Software Requirements and Prototyping, and Software Design. Each of these courses has a substantial group project component. The Software Project Lab course represents the culmination of the program, where students, working in teams, apply the technical and project management skills from the courses to the development of a software system. The project lab is intended to simulate industrial conditions and is discussed in more detail by McKeeman [McK87].

To be accepted into the SWSE program, students must have satisfactory GPA and GRE scores, preferably at least one year's industrial experience, and undergraduate courses or equivalent knowledge in block-structured programming, data structures and algorithms, computer organization, and discrete mathematics.

### **3 THE INFORMATION TECHNOLOGY PHD PROGRAM**

Our PhD program is administered through the Dean's office of SITE. The students obtain a PhD in Information Technology (INFT) by taking six MS-level breadth courses from at least three of the departments

in SITE, six depth INFT PhD-level courses, other supporting coursework and seminars, and writing a PhD dissertation. There are currently over 300 students in this program. The ISSE and CS faculty contribute to the PhD program by teaching several INFT courses and by advising and serving on committees of PhD students.

The following software engineering courses have been offered as part of the PhD program: Software Engineering Seminar, Software Productivity, Topics in Software Requirements, Experimental Methods in Software Engineering, Software Maintenance and Reuse, Software for Critical Systems, Software Analysis and Design of Real-time Systems, Concurrent Object-oriented Systems, and Software Reuse.

Software engineering has become one of the major focus areas in our PhD program, with 8 of the last 21 PhDs awarded being in the software area. The topics have primarily been in the areas of requirements engineering, prototyping and domain analysis. Plans are currently underway to formalize this interest by creating several tracks within the program, one of which would be software engineering.

## **4 INTEGRATION WITH OTHER GRADUATE PROGRAMS**

The Software Systems Engineering program at George Mason University is closely integrated with three other degree programs: the masters degree programs in Computer Science, Information Systems, and Systems Engineering. Together, the programs allow students to select a focus of study from a spectrum that ranges from traditional computer science topics through the technical engineering issues of constructing software systems to the business, management, and use of information systems.

### **4.1 The Certificate in Software Systems Engineering**

One aspect of the integration between the four programs is the Certificate in Software Systems Engineering. The certificate program was created to provide software engineering education to MS students in related disciplines. It is mostly pursued by students in the computer science, systems engineering or information systems masters programs who take five SWSE courses, consisting of the MS-SWSE core courses with the exception of the project laboratory, in addition to the core of their masters program and in place of other electives. Partially to make it easier for students to complete the certificate as well as their own requirements, the Computer Science Department has several of the SWSE courses cross-listed as their own courses. The certificate may also be pursued by students who already have an MS degree in a related discipline.

## 4.2 The Foundation Courses

We are currently considering standardizing foundation requirements for the Software Systems Engineering, Computer Science, and Information Systems programs. Although there are undergraduate courses that can satisfy a student's need for missing foundation courses, such courses are not a good fit for our typical students, many of whom return for graduate school after an extended experience in the workplace. Problems with undergraduate foundation courses include the ability of graduate students to cover foundation work at a faster pace than undergraduates, a lack of tailoring of the material to the specific needs of our program, an inability of employer-sponsored students to obtain support and tuition-reimbursement for courses that are not offered at the graduate level, and the fact that most of our students need to take courses at night so they can work during the day. To meet the needs of students who are deficient in one or more foundation courses, graduate level foundation courses either exist or are planned for each of the prerequisite subjects: block structured programming languages, data structures and algorithms, computer organization, and discrete math.

# 5 SOFTWARE SYSTEMS ENGINEERING COURSES

Our masters curriculum has six core courses, and we offer several electives. Each of these courses is described below, with emphasis on what we actually teach in the courses, rather than the formal catalog descriptions.

## 5.1 The Core Courses

**SWSE 619: Software Construction.** This course presents the concepts of information hiding, data abstraction, concurrency, and object-oriented software construction, as well as general issues relating to the software lifecycle. The course focuses on reactive software, and uses sequence diagrams and state diagrams (including some basic concepts of automata theory) to describe software. We have typically used Ada as a vehicle to present the concepts, and the course usually includes several Ada programming projects. The class uses Sanden's text [San93b], and presents design and implementation issues at the small to moderate level.

**SWSE 620: Software Requirements and Prototype.** This course presents an in-depth study of the methods, tools, notations and validation techniques for the analysis and specification of software requirements. It includes a group project. SWSE 620 has been taught with an emphasis on prototyping and with an emphasis on object-oriented analysis according to Rumbaugh's OMT [RBP+91]. The main requirements

text is Davis's [Dav93]. A variety of prototyping tools are used including PC Access, Oracle/C, Hypercard, Toolbook and Expert Problem Solving System (EPSS).

**SWSE 621: Software Design.** This is a course in concepts and methods for the architectural design of software systems of sufficient size and complexity to require the effort of several people for many months. We introduce fundamental design concepts and design notations. Several design methods are presented and compared, with examples of their use. The course text is by Gomaa [Gom93]. Students undertake a term project working in small groups to design a relatively complex software system using one design method.

**SWSE 623: Formal Methods and Models in Software Engineering.** SWSE 623 covers formal mechanisms for specifying, validating, and verifying software systems. The course presents program verification using Hoare's method and Dijkstra's weakest preconditions (based on Gries's text [Gri81]). Formal specification is taught through algebraic specifications and, to a larger extent, abstract model specifications. The treatment of algebraic specifications is based on the approach of Liskov and Guttag [LG86]. The treatment of abstract model specifications currently employs the specification language Z (using Potter, Sinclair and Tills' text [PST91]) and covers both initial specifications and refinement towards implementation. The course addresses the integration of formal methods with existing programming languages such as Eiffel and Ada, the application of formal methods to a variety of topics such as requirements analysis, testing, safety analysis, and object-oriented approaches, and arguments for using formal methods, including appropriate conditions for their application and trade-offs with less formal techniques. The course presently includes the formal specification of a simple file system as an exercise designed to show how formal methods can disambiguate informal specifications and separate genuine requirements from implementation decisions.

**SWSE 625: Software Project Management.** SWSE 625 covers lifecycle and process models, process metrics, and planning for software projects. It includes mechanisms for monitoring and controlling schedule, budget, quality and productivity, as well as leadership, motivation, and team building. The course emphasizes computer-based tools to support software project management.

**SWSE 626: Software Project Laboratory.** In this course, students are involved in the analysis, design, implementation and management of a software system project. They work in teams to develop or modify a software product, applying sound software engineering principles. Although the project varies each semester, the purpose of the project has usually been to develop a software system either according to ADARTS [Gom93] or Entity-life Modeling [San93b]. SWSE 626 is taken almost exclusively by MS-SWSE students. One section is offered each semester with enrollments from 7 to 25. Occasionally, students may work on the same project in SWSE 621 and SWSE 626. While this is desirable, we cannot systematically rely on multi-semester projects since the students do not progress through the program as one class.

The project laboratory is ideally suited for students entering graduate school immediately from college and for junior software engineers with little previous opportunity to follow a project through several development phases. It is also popular among students who manage to put a good team together and produce a software product that far exceeds course requirements.

The two core courses on software construction and design plus the project lab permit us to address fairly advanced design issues. In particular, the project lab has made it necessary for us to introduce a common programming language. We have selected Ada, which has been a good choice particularly in a geographical area where many students have a background in systems development for government and defense [San93a].

## 5.2 Elective Courses

**SWSE 630: Software Engineering Economics.** SWSE 630 covers quantitative models of the software lifecycle and cost-effectiveness analysis in software engineering. Some of the specific topics are multiple-goal analysis, uncertainty and risk analysis, software cost estimation, software engineering metrics, and quantitative lifecycle management techniques.

**SWSE 631: Object-Oriented Software Development.** SWSE 631 covers principles of object-oriented design, development, and programming in Eiffel, based on Meyer's text [Mey88]. It includes relationships between object-oriented design concepts and software engineering principles, techniques of object-oriented design and programming, and applications of object-oriented techniques. It also includes emphasis on design, evolution and reuse through supplemental readings.

**SWSE 632: User Interface Design and Development.** This design course studies the role of the human in the design and implementation of software, and prepares students to design and evaluate the quality of interfaces between computer software and human users. The course presents theories of human-computer interaction, including human cognitive limitations, syntactic versus semantic knowledge, transitionality, and the "outside-in" design approach. It also presents guidelines for designing various types of computer interfaces, including command interfaces, menus, desktop views, and graphic interfaces. The course currently uses Shneiderman's text [Shn92], with supplemental material derived from Brown and Cunningham [BC89], and recent literature. Students evaluate several user interfaces, and either write a paper covering an interface topic or design and implement a user interface.

This course has appealed to SWSE students, CS students, and students from a variety of disciplines, including education and visual information technology (from the Arts department). Part of the challenge

of this course is to balance the interdisciplinary nature of the subject with the technical aspects of software engineering, which are often quite difficult for students in less technical majors.

**SWSE 635: Software Testing and Quality Assurance.** SWSE 635 covers software testing and quality assurance techniques at the module, subsystem, and system levels. Topics include control and data flow coverage, mutation analysis, fault-based testing, symbolic evaluation, test specifications, test generation, category-partition testing, inspections, requirements-based testing, statistical testing, and reliability assessment. The course uses Beizer's book [Bei90], heavily supplemented with research papers from the recent literature. Students plan for and carry out the testing of an example software system. The course has a technical focus, and many of the management-oriented concerns are deferred to SWSE 625.

**SWSE 720: Advanced Software Requirements.** This course covers the current research and the current applications of requirements engineering. It focuses on critical problems in software engineering and discusses how their resolution might enhance the quality and productivity of real software and system developments in industry.

**SWSE 721: Advanced Software Design Methods.** This course studies advanced software design methods for large-scale software systems, including concurrent, real-time, and distributed systems. Students apply one or more of the methods to the design of a relatively complex software system. This course complements SWSE 621 for students particularly interested in software design. With SWSE 621 focusing on DARTS and ADARTS, SWSE 721 has been based on entity-life modeling (ELM) according to Sanden [San93b].

### 5.3 Physical Laboratories

The Software Systems Program has access to computer facilities provided by the University and the School, as well as a laboratory of networked Sun workstations provided by the Center for Software Systems Engineering. These suns are currently connected to the ISSE department file server, with the internet subdomain of `isse.gmu.edu`. The University provides DEC machines running both Unix and VMS for general use. SITE provides a networked lab of HP X-terminals that we have used in courses, particularly for the Ada programming projects in SWSE 619. A surprising result of our mix of students is that university-provided equipment is in less demand than at more traditional universities. The majority of our students are working full-time and do not come to campus every day, so they prefer to use their own equipment at home or at work when possible.

## 6 CURRENT STATUS AND STATISTICS

Student enrollment in the two programs has grown rapidly. Approximately 83 students are currently enrolled in the Masters program. The fact that many of these students are working professionals in the field has impacted many aspects of our program, both positively and negatively. As compared to traditional master's students at other universities, most of our students are older and, because of the extra burdens of completing an advanced degree with the responsibilities of jobs and often families, are more determined to succeed. They are also more cognizant of the value of an education and aware of the difficulties. From a technical perspective, the students tend to be better writers (a result of their experience and maturity), but because their undergraduate degrees are often 5 to 15 years old, our students are often less prepared technically than more recent graduates. Some of the material that is currently being taught in data structures courses, such as data abstraction, information-hiding, and data structures and algorithms for advanced graph and tree structures is known to only a fraction of our incoming students. In spite of these problems, we are constantly impressed by their hard work, dedication and enthusiasm.

The software engineering courses also attract many students from other Master's programs, in particular Computer Science and Information Systems, who take the courses as electives counting towards their degrees. Some students switch to the Masters program in Software Systems Engineering. Many of the students from related disciplines take the Software Systems Engineering certificate.

### 6.1 Enrollment Statistics

The intentions of the many students in software engineering courses can be derived only in part from enrollment data, and so we conducted a survey of students taking software engineering courses during the Spring 1993 semester. The survey results reflect school-wide interest in the software engineering curriculum. A total of 248 questionnaires were returned (some duplicate entries were rejected and are not counted in the 248 replies). The results were tabulated by the type of degree being pursued and by whether or not students intended to pursue a Software Systems Engineering Certificate. The results are summarized in Table 1 below.

<i>Degree Being Sought</i>	<i>Number of Students</i>
Masters in Software Systems Engineering	<b>83</b>
SWSE Certificate Students	
SWSE Certificate Only	10
CS Masters and SWSE Certificate	18
IS Masters and SWSE Certificate	16
Other Masters and SWSE Certificate	4
Total SWSE Certificate Students	<b>48</b>
Other Students	
CS Masters only	64
IS Masters only	22
Other Masters only	20
Other responses	11
Total Other Students	<b>117</b>
Total	<b>248</b>

Table 1: Majors of Students Enrolled in SWSE Courses in Spring 1993.

The table shows that about one third of the students enrolled in SWSE courses are pursuing a Masters in Software Systems Engineering. In addition, about a fifth of the respondents are pursuing a certificate in some form or other. Putting the previous two observations together, we see that just over half of the students in software engineering courses are pursuing a Certificate or Masters in Software Systems Engineering. Correspondingly, the remaining students find software engineering courses interesting as electives, even though such students do not plan either a Certificate or a Masters in Software Systems Engineering.

## 6.2 Course Offerings

Up to and including the Fall 1992 semester, each of the core courses was offered in both the Fall and Spring semesters, and the electives were offered once a year, with typical course enrollments of between 35 and 50 students. However, a sudden growth in enrollments was experienced last fall, forcing us to accept over 60 students in some classes. Consequently, in the Spring '93 semester, two sections were offered for three of the core courses. In Fall '93, two sections will be offered for four of the five courses that compose both the software engineering core and the software engineering certificate. The courses in Spring '93 were capped at 40, and with two sections offered in several cases, it was thought that this would more than compensate for student demand. To our surprise, most of the sections were filled to maximum capacity.

SWSE 619 has become a very popular course with students in various majors; for example, over half of the students taking SWSE 619 in Spring 1993 were computer science majors. We are offering an additional section of 619 in the summer.

The first graduates of the Certificate program were in Spring 1989, while the first graduates of the MS program were in Spring 1990. Up through May 1993, 45 Master's students have graduated in Software Systems Engineering.

## 7 DISCUSSION

The teaching faculty consists of four full time professors in the ISSE department and one professor who has a joint appointment with the CS and ISSE departments, assisted by one or two research professors (funded primarily by research contracts) and several adjunct professors. The department is fortunate to have a pool of dedicated, experienced and highly capable adjunct professors, without whom the SWSE program would be compromised. In addition to the core courses, there are usually two or three electives taught each semester.

Several of the software engineering courses offered in the Masters and PhD programs have resulted in text books being written by GMU faculty. These include books on Software Systems Engineering [SP90], Software Construction [San93b], and Software Design [Gom93].

The Masters and Certificate programs in software engineering have grown rapidly over the last five years, even though budget cutbacks due to the difficult economic climate in the state of Virginia has actually resulted in two less faculty than originally planned. Because of budgetary constraints and because there are comparatively few full-time students, the pool of potential graduate teaching assistants (GTAs) is small. In addition, because of the fact that the group projects in Software Requirements, Software Design, Software Project Management, and Software Project Lab require considerable software engineering expertise to manage, the contribution that can be made by GTAs is limited. Thus, teaching an effective software engineering program is particularly demanding on the faculty, requiring considerable faculty time for meetings with students and grading project deliverables in addition to a regular lecturing schedule.

The ISSE department recently proposed an undergraduate program in Information and Software Systems Engineering, combining the product (software systems) and process (software engineering) perspectives of large scale software system development. Unfortunately there are no resources available for such a program in the foreseeable future, as the department is already understaffed in both masters programs it offers.

## 8 CONCLUSIONS

The Master of Science program in Software Systems Engineering is now in its fifth year at George Mason University. There has been rapid growth in student enrollment in software engineering courses, by MS students in software engineering as well as graduate students enrolled in other programs. The Certificate program has proved to be an effective vehicle for channeling students in other degree programs into a specialization in software engineering. There is also considerable interest in software engineering at the PhD level.

Our program specifically meets the 7 criteria listed by Freeman [Fre87], originally from Freeman et al. [FWF76]:

1. Be based on the five content areas of computer science, management, communication, problem solving, and design.
2. Be flexible so that they can change easily and be adapted to substantive developments in the field.
3. Be based on computer science and be viewed as “applied computer science”.
4. Prepare students to push forward the boundaries of knowledge and techniques, not just apply what is already known.
5. Include a large amount of realism and practical work.
6. Provide for multiple implementations, dependent upon career objectives and backgrounds of students and upon the academic home of the program.
7. Build on existing curricula to the extent possible.

These criteria were appropriate when originally proposed in 1976, and still apply today. In one form or another, these criteria help determine all the decisions we make about our program.

One of the most challenging aspects of our curriculum is the unusual mix of students that we teach. The fact that most of our students are experienced professionals is an element that is likely to be common to future software engineering programs. Although our location exacerbates the challenges, software engineering educational programs will continue to be populated, and should be populated, by part-time professionals with experience. We have found that this results in a large number of unusual situations, from students who are more dedicated to their education and appreciative of the principles we teach, to students who, because of professional obligations, must resort to submitting assignments and even exams by fax. It is our belief that the traditional educational systems are undergoing drastic changes, and the non-traditional educational choices that we have made in our program provide models for appropriate advanced education into the next century.

## 9 ACKNOWLEDGEMENTS

Our program was initiated through a proposal by Richard Fairley to the Commonwealth of Virginia, and we gratefully acknowledge his groundbreaking efforts, as well as the efforts of Peggy Brouse, Al Davis, Carolyn Davis, Peter Freeman, Jorge Díaz-Herrera, Ken Nidiffer, Larry Kerschberg, Gene Norris, Jim Palmer, and Andy Sage, who have all contributed to the program's success.

## References

- [ADF<sup>+</sup>91] P. Ammann, A. Davis, R. Fairley, H. Gomaa, and B. Sanden. Graduate programs in Software Engineering at George Mason University. In *Proceedings of the 1991 SEI Software Engineering Education Workshop*, Austin TX, May 1991.
- [Ard87] M. Ardis. The evolution of Wang Institute's Master of Software Engineering program. *IEEE Transactions on Software Engineering*, SE-13(11):1149–1157, November 1987.
- [BC89] J. Brown and S. Cunningham. *Programming the User Interface*. John Wiley & Sons, New York NY, 1989.
- [Bei90] B. Beizer. *Software Testing Techniques*. Van Nostrand Reinhold, New York NY, 2nd edition, 1990.
- [Dav93] A. Davis. *Software Requirements: Objects, Functions, and States*. Prentice-Hall, Englewood Cliffs, NJ, 1993.
- [Fai88a] R. Fairley. Post-mortem analysis of software engineering at Wang Institute. *ACM SIGSOFT Notes*, 13(2):41–47, April 1988.
- [Fai88b] R. Fairley. The software engineering programs at George Mason University. In *Proceedings of the SEI Conference on Software Engineering Education*, Fairfax VA, April 1988. IEEE Computer Society Press.
- [Fre87] P. Freeman. Essential elements of software engineering education revisited. *IEEE Transactions on Software Engineering*, SE-13(11):1143–1148, November 1987.
- [FWF76] P. Freeman, A. I. Wasserman, and R. E. Fairley. Essential elements of software engineering education. In *Proceedings of the 2nd International Conference on Software Engineering*, pages 116–122. IEEE Computer Society Press, 1976.
- [Gom93] H. Gomaa. *Software Design Methods for Concurrent and Real-Time Systems*. Addison-Wesley, Reading MA, 1993.
- [Gri81] D. Gries. *The Science of Programming*. Springer-Verlag, New York, 1981.
- [LG86] B. Liskov and J. Guttag. *Abstraction and Specification in Program Development*. The MIT Press, Cambridge, MA, 1986.
- [McK87] W. M. McKeeman. Experience with a software engineering project course. *IEEE Transactions on Software Engineering*, SE-13(11):1182–1191, November 1987.

- [Mey88] B. Meyer. *Object-Oriented Software Construction*. Prentice-Hall International, Englewood Cliffs, NJ, 1988.
- [PST91] B. Potter, J. Sinclair, and D. Till. *An Introduction to Formal Specification and Z*. Prentice Hall, New York, 1991.
- [RBP<sup>+</sup>91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-oriented Modeling and Design*. Prentice-Hall, Englewood Cliffs, NJ, 1991.
- [San93a] B. Sanden. An Ada-based, graduate software-engineering curriculum at GMU. In *Proceedings of the 7th Annual ASEET Symposium*, pages 119–124, Monterey CA, January 1993.
- [San93b] B. Sanden. *Software Systems Construction with Examples in Ada*. Prentice-Hall, Englewood Cliffs, NJ, 1993.
- [Shn92] B. Shneiderman. *Designing the User Interface*. Addison-Wesley, Reading MA, 2nd edition, 1992.
- [SP90] A. P. Sage and J. D. Palmer. *Software Systems Engineering*. John Wiley & Sons, New York NY, 1990.