Rochester Institute of Technology Bachelor of Science in Software Engineering CSEE&T Hall of Fame Nomination

James R Vallino
Department of Software Engineering
Rochester Institute of Technology
Rochester, NY, USA
J.Vallino@se.rit.edu

***Abstract*— In the fall of 1996, the Rochester Institute of Technology launched the first undergraduate software engineering program in the United States. The culmination of five years of planning, development, and review, the program was designed from the outset to prepare graduates for professional positions in commercial and industrial software development. From an initial class of 15, the ABET-accredited program has grown steadily over the intervening years until today the student body numbers over 580 undergraduates.**

Keywords—software engineering education, RIT software engineering, undergraduate software engineering

I. Introduction

In the fall of 1996, the Rochester Institute of Technology (RIT) launched the first undergraduate software engineering program in the United States [1-3]. In May 2001, the program was the first to award BS in Software Engineering degrees. To date, the program has nearly 870 graduates. The culmination of five years of planning, development, and review, the program was designed to prepare graduates for professional positions in commercial and industrial software development. The program, in a separate Department of Software Engineering, has the independence and flexibility necessary to ensure its integrity over time.

The program's focus on educating professional, practicing software engineers is best illustrated by the required year of co-operative education. At the end of the five year program, the students have both solid academic preparation and significant practical experience. Our graduates are in high demand, as they are prepared to define, design, develop and deliver quality software systems.

II. Curriculum

A. *Inspiration*

The inspiration for starting the program came when one of the program's founders, Mike Lutz, was on a two-year industrial leave from RIT. During this leave he led software development teams and was responsible for interviewing, hiring, and mentoring recent college graduates. He noticed that, by and large, these graduates had solid preparation in basic computing theory and technology, but they lacked the background necessary to be effective when working on large, complex, industrial quality systems. That realization resulted in the RIT software engineering program with a curriculum addressing the knowledge and abilities many graduates of traditional computing programs lacked. The curriculum was designed from a blank sheet of paper instead of making tweaks to an existing program [4]. We refer to this as our Manifesto for Software Engineering Education [5].

B. *Four Curricular Themes*

The curriculum has four defining themes which have remained constant throughout the program's evolution over its twenty-one years. These themes are software design, software process, teamwork, and communication. The first two have an equal weighting in terms of course content between "design" courses and "process" courses. The latter two themes, often considered soft skills, cross-cut through every course in the curriculum. This is in contrast to most engineering and computer science programs where any discussion of process, teamwork, and communication is relegated to one or two capstone design courses during the senior year. In our program, the two-term software engineering senior project is where our students demonstrate the full range of their skills. The course delivers no new content. Student teams are fully prepared to use skills in these four theme areas to deliver working software systems to their industrial and non-profit project sponsors.

C. *Software Design*

Our students' primary exposure to programming per se is in the introductory computer science sequence. While we do discuss programming techniques in software engineering courses, we focus on it as just one aspect of product delivery. This gives us room to emphasize more significant issues of modeling and design [6] drawing the students to focus on the higher level component relationships. Our design courses discuss the engineering aspects of designing software for web-based systems, real-time and embedded systems, cloud-based systems, enterprise information systems, and secure systems.

D. *Software Process*

When we were creating the curriculum, the notion of teaching professionalism as encapsulated in a disciplined process was prominent in our thinking. Process is not, as some claim, the be-all-and-end-all of software engineering, but it does provide the frame within which software development takes place. In our curriculum, process is as important a pillar as software design. This does not mean, however, that we impose one dogmatic approach to process - indeed, we ensure that students are familiar with many process approaches, from strictly planned to the more adaptive agile approaches. Our process courses include: Software Process and Project Management, Software Process and Product Quality, Software Testing, and Trends in Software Development Processes.

E. *Teamwork*

Working on teams to solve problems is a hallmark of our software engineering program. Indeed, with two exceptions (the Personal Software Engineering and the Mathematical Models of Software courses), all of the software engineering courses incorporate team projects as a graded component. For that portion of the grade based on team activities (40 – 50%), teams receive grades as a whole. However, we also assess each team member in terms of his or her contributions to the team, adjusting each individual's grade based on instructor observations, version control logs, and peer evaluations. Through the course of their program, the typical SE students will work on over twenty different teams.

F. *Communication*

There is a clear need for multi-modality communication within engineering disciplines. In addition to a required Professional Communication course, our students see this continually throughout their program. Multiple times within each course, students will make presentations about their work progress either individually or as part of a team. They also prepare documents in the form of requirements, design, risk management, or test plans in every class.

III. Accreditation Success

Program faculty participated in discussions for formulating and interpreting initial criteria for ABET EAC accreditation of software engineering programs [7,8]. The program has gone through four successful rounds of accreditation review. It has the distinction of having the earliest graduates from an accredited program starting with the 2001 class. In the last two reviews, the Program Evaluator recommended that the Self-Study report be displayed at the spring ABET Symposium as an exemplar for excellence in assessment process and preparation of the self-study.

IV. Employment Success

In most years, software engineering students report the highest average hourly co-op and median full-time wages compared to all other undergraduate majors at RIT. Our placement rate is over 90% at graduation. Co-op employment evaluations provide other anecdotal evidence of our students' value to their employers. An engineering manager in an aerospace company commented that our students have a strong focus on capturing requirements and system modeling. An engineering vice-president noted that our graduates match up favorable against some software engineers with five years of experience.

V. Conclusion

Twenty-one years ago when we created the first undergraduate software engineering program in the US, we gambled that if we built it, they would come, which includes an attraction for both students and employers alike. Our track record of continual program growth and over 90% placement of graduates demonstrates that the gamble paid off. We believe that our program, with its engineering design, software process, teamwork, and communication themes, provides students who seek a career in software development with a set of skills better tailored to what is needed to excel not only as an entry-level software engineer but also for growth throughout their career. Subsequent undergraduate software engineering programs have used our curriculum as their "design pattern". Based on the RIT program's distinguishing characteristics, success, and influence on software engineering education, we believe that it is worthy of being elected to membership in the CSEE&T Hall of Fame.

Acknowledgment

We want to acknowledge the nearly 870 undergraduate students who have gone through our program. They worked hard to obtain the skills that our curriculum taught them. We particularly appreciate the students who stuck with us in the earlier years when we were figuring out software engineering as an undergraduate discipline, and those who went through the turmoil of a quarter to semester calendar conversion.

References

[1] M. J. Lutz and J. F. Naveda, "The road less traveled: a baccalaureate degree in software engineering," In Proceedings of Conference on Software Engineering Education and Training. (1997).

[2] Naveda, F. and Lutz, M. "Crafting a baccalaureate program in software engineering," In Proceedings of Twenty Eighth SIGCSE Technical Symposium on Computer Science Education. San Jose, CA, (1997).

[3] F. Naveda, M. J. Lutz, J. R. Vallino, T. J. Reichlmayr, and S. A. Ludi, "The Road We've Traveled: 12 Years of Undergraduate Software Engineering at the Rochester Institute of Technology," Proceedings of International Conference on Information Technology: Next Generations, Las Vegas, NV, April 2009.

[4] RIT Software Engineering Undergraduate Program, http://www.se.rit.edu/undergraduate

[5] M. J. Lutz, J. F. Naveda, and J. R. Vallino, "Undergraduate software engineering," Commun. ACM, vol 57, pp. 52-58, August 2014.

[6] J. Vallino, "If you're not modeling, you're just programming: modeling throughout an undergraduate software engineering program," In Proceedings of the 2006 International Conference on Models in Software Engineering, 291–300, October, 2006.

[7] M. J. Lutz, "Accreditation Criteria for Programs in Software Engineering." David Lorge Parnas Symposium, 2001 International Conference on Software Engineering. Toronto, CA. May, 2001.

[8] J. McDonald, M. Sebern and J. Vallino, "Software Engineering Program Accreditation in the United States." In Software Engineering: Effective Teaching and Learning Approaches and Practices, H. Ellis, S. Demurjian, and J. F. Naveda, (eds), Information Science Reference, 2008.