Developing a Process for Software Engineering Curriculum Modernization

Dr. Emily Marasco, University of Calgary

Dr. Emily Marasco is an Assistant Professor (Teaching) of software engineering and the SSE Teaching Chair in Engineering Education Innovation – Digital Transformation. She is the founder of the ADAPT Lab: Advancements in Digital Applications and Pedagogical Transformation. Her current research and teaching interests are in the area of learning engineering, including the use of gamification, blended learning, and interactive digital learning tools as methods for enhancing creativity within software and computer engineering. Her pedagogical initiatives for digital transformation in education include digital and AI literacy, integration of cognitive diversity, and accessibility best practices.

Dr. Marasco is active as a science communicator and outreach speaker in the local education community. She has been recognized as the 2018 ASTech Outstanding Leader of Tomorrow and received the 2016 Claudette MacKay-Lassonde Graduate Award for women in engineering. She was most recently recognized as one of Calgary's 2019 Top 40 Under 40 recipients.

Ms. Milana Hayley Grozic, University of Calgary; The University of British Columbia

Ms. Milana Grozic (she/her) is a second year psychology major at The University of British Columbia. Her research attempts to merge the fields of engineering and psychology - focusing specifically on engineering education. Her passion for the human psycho extends far beyond psychology and she is proud to bring her unique perspectives into engineering education research.

Yves Pauchard, University of Calgary
Dr. Mohammad Moshirpour, University of Calgary

Dr. Mohammad Moshirpour is an instructor of Software Engineering at the Schulich School of Engineer, University of Calgary. His research interests are the area of software architecture, software requirements engineering, design, implementation and analysi

Developing a Process for Software Engineering Curriculum Modernization

Abstract

The Schulich School of Engineering has recently undertaken staged redesign and implementation of a new software engineering curriculum. Stakeholders were asked to consider a set of formulated questions for their topic and related list of courses. Consultation comments, suggestions, and previous feedback were evaluated and incorporated into the proposal. The proposed curriculum changes were rolled out in a staged approach. The rollout of the new second year curriculum started in Fall 2022 with the new third year curriculum beginning in Fall 2023. An initial survey was conducted to evaluate student feedback on course content and experience. Using a quasi-experimental post-test only design, students who experienced both the new and/or the old curriculum were asked to rank their academic experience including factors such as course content, workload, stress, engineering identity, graduate attributes, and more. This paper will outline and discuss the process that was undertaken to evaluate, design, consult, implement, and now re-evaluate multi-year curriculum changes, including a continual improvement process.

Motivation

As software systems and related technologies have become increasingly complex, the demands placed on software engineering education have grown [1, 2]. Current priorities in software engineering pedagogy include experiential learning and alignment with modern, industry-relevant practices to solve problems [1, 2, 3]. Like many institutions, the University of Calgary's Schulich School of Engineering has seen immense growth in the software engineering program, with increased enrollment in recent years. This fast growth has resulted in increased class sizes and number of lecture sections, rapid hiring of new faculty, and an urgent need for additional classroom space. The expanding student interest and need to reimagine the program to adapt to the ever-changing technological landscape of the 21st century led to an opportunity to revise and modernize the software engineering curriculum.

Creating change within an academic program requires an understanding of change management, academic freedom, research-informed pedagogical practices, and more. Kolmos, et. al discuss how varied response strategies for curriculum change in engineering require coordination and consideration at different levels of the institution [4]. For example, small changes maybe championed by individual academics while strategies may be mapped and coordinated by program leaders or even require institutional-level direction.

The changes within this curriculum redesign relied on a taskforce consisting of academics within the software engineering program who had previously championed pedagogical change within their own teaching and learning practice. The taskforce was created to analyze potential areas of curriculum revision. The members of the taskforce included teaching-stream faculty members engaged in pedagogical research as well as the software engineering program director.

The goals for the curriculum revision were identified to be:

1. Content modernization to reflect changing needs and practices in software engineering
2. Cohesive alignment of vertical progression that links each year of study
3. Increased integration of course concepts and collaborative pedagogy
4. Keep current with leading-edge technologies and approaches
5. Student-focused to provide skills and knowledge needed to thrive in industry or graduate programs
6. Raise department profile and increase competitiveness with other software engineering programs

The degree program objectives were identified as a) to graduate future software engineers as practitioners, researchers, developers and collaborators, b) to integrate fundamental knowledge and applied skills, c) to develop lifelong learning capacity through real-world projects and industry-based training, and d) to train well-rounded software engineers adept in industry-relevant professional skills.

This paper will detail the development and implementation of the consultation and redesign process, including final curriculum content changes and related delivery recommendations.

Consultation Process

The taskforce consulted several subcommittees and stakeholder groups to adequately assess the changing landscape of software engineering. These stakeholder groups included faculty members within the department, faculty members in related departments who may be impacted by the course changes, industry advisors, and faculty administrators. Consultation was done in a staged approach that allowed iterative revisions while still centering discussion on the previously mentioned curriculum redesign goals.

Early conversations with engineering industry stakeholders and community groups emphasized the incorporation of key elements, such as project-based learning and industry involvement, as well as the integration of professional competencies including knowledge of sustainability and team software processes [3].

A set of curriculum evaluation guidelines were created by the taskforce to explain the goals and motivation behind the redesign, as described above. These were provided to each stakeholder group to help provide direction. Several disciplinary subcommittees were also created to focus suggestions on how the curriculum content may not be meeting the needs of students or industry. As students within the faculty complete a common core first-year, disciplinary content was focused on introductory 2nd year competencies and expanding core skills in 3rd and 4th year. Additional committees were added to analyze the existing hardware and signals content within the curriculum, as well as to explore whether machine learning content needed to be added.

The final subcommittee divisions were:

1) 2nd Year Software Development Principles
2) 3rd/4th Year Software Development Principles
3) Hardware for Software Engineers
4) Machine Learning Integration
5) Signals for Software Engineers

Stakeholders were also informed that to keep up with changing technology and industry needs, space must be made in the existing curriculum for new topics. Evaluation was to remain student-focused while considering how the curriculum could provide the skills and knowledge needed to thrive in industry or graduate programs. Each subcommittee was assigned specific courses/content to evaluate, and were asked to consider the following questions:

- What topics must be covered in these areas?
- What needs to be updated in these courses and do any changes need to be made to course descriptions?
- What topics are no longer relevant or could be removed?
- To keep up with changing technology and industry needs, space must be made in the existing curriculum for some new topics
- How could these courses potentially be integrated?

All stakeholders were also asked to consider the existing capstone course and how improvements could be made. A member of the taskforce attended each consultation session to track the discussion so that recommendations could be collated and integrated across all groups.

Based on the aggregate consultation outcomes, the taskforce highlighted the following priorities for action:

- Highlight course differences between the software engineering program and the computer science program, including the application of software design principles and modern project management
- Acknowledge the changing nature of software engineering and how content may need to adapt rapidly
    - Course descriptions should not focus on specific frameworks, languages, libraries, etc. but rather on tooling, processes, skills
- Integrate a project-based learning approach to include technical knowledge across multiple courses
    - Highlight the practical relationships between different subjects (such as software architecture and databases) while emphasizing hands-on learning
- Incorporate contemporary topics, tools and practices
    - Implement multidisciplinary projects to train our students in using software solutions as an effective tool in different domains and foster their abilities in problem-solving in various fields
    - Incorporate cloud, DevOps, machine learning and AI into student projects
- Increase industry involvement and sponsored projects
    - Adapt hackathons as an effective tool for training and evaluation
    - Foster a community of learning

- Map each course and curriculum content as part of a cohesive progression allowing all faculty to better understand how their own course is situated within the program
    - Faculty members need to continue content dialog to ensure on-going smooth transitions between prerequisite, corequisite, and subsequent courses
- Increase development of professional skills within technical content to build a portfolio of applicable industry skills that align with the Canadian graduate attributes for engineers [5]
    - Knowledge of sustainability, including case studies to demonstrate applications in ethical development, equity and accessibility issues (e.g. web scraping)
    - Future-proofing
    - Economics
    - Project management and team software processes
    - Communication skills
    - Conflict resolution
    - Individual software processes, resiliency, self-reflection, self-assessment
    - Revision control and use of tools
    - Innovation and creative capacity
    - Entrepreneurial mindset

Overall, students and faculty both wanted to see more diversity in senior technical electives. These courses also allow students to specialize in specific areas of interest since it is not possible to cover the immense breadth of software engineering in a limited timeframe. Students, faculty, and industry representatives also identified a gap in security content within the program. Rather than introduce a separate course, it was discussed that security concepts need to be introduced across multiple courses. For example, a single front-end course could offer an opportunity to discuss security in web development. A more in-depth technical elective could offer an advanced look at security concepts. Faculty members also felt that students need to be introduced to revision control and CI, with testing and deployment concepts introduced sooner in the program.

The taskforce used the above recommendations to analyze gaps in the existing program. New courses were proposed as well as the elimination or adaptation of other content, to be discussed in the next section. Each proposed professional competency was mapped to specific courses within the program. Vertical and horizontal connections were mapped between calendar descriptions and aligned with prerequisite requirements to ensure appropriate continuity. The majority of changes were focused on the second and third year content to help prepare students for their internship year. First year was not modified due to the common core nature of the program, and the final year remained focused on technical elective choices and the capstone design experience.

Early proposal drafts and feedback were discussed at multiple departmental meetings (including student feedback), as well as faculty-level meetings. Industry members across varied disciplines continued to provide consultation feedback and letters of support as the new curriculum went through the university governance procedures.

Staged Implementation

The proposed curriculum changes were rolled out in a staged approach. Changes to the second-year content were first applied for 2022-2023, with changes to the third-year content beginning in 2023-2024. The first cohort of students who were enrolled in the new curriculum are now finishing their third year of studies and are preparing either for a 12-to-16-month internship, or to continue directly into fourth year.

The major changes to the curriculum can be summarized through the following themes and actions.

New courses added to highlight engineering design and professional skills:
Professional skills are currently embedded primarily in the final capstone course. While some earlier courses include teamwork and project management, many courses are already full with technical concepts that must be covered. The introduction of courses in second and third year that combine industry-relevant professional tools and skills will help to build student confidence in areas such as the use of revision control tools, entrepreneurial thinking, ethical case studies, and software development practices. These consecutive courses will better prepare students for their internships and capstone design projects.

Adaptations to differentiate from computer science content:
New courses were created in place of students enrolling in existing computer science courses. Within the computer science program facing similar challenges in terms of enrollment and rapid expansion, separating more of the curriculum content allows both programs to better tailor content towards their specific disciplinary applications while reducing pressures within the registration and scheduling systems. The new applied content includes the implementation of algorithms and data structures, the bridging of concepts between computer organization and the functionality of processors, discussions around operating system security, distributed systems within engineering, and data management for engineering applications.

New courses to address disciplinary gaps:
The consultation and mapping process revealed a disciplinary gap around full-stack development. The recommendations led to the development of a new course that provides students with a comprehensive full-stack training with an integrated understanding of the challenges of software development lifecycles.

The faculty-wide focus on data and machine learning aligned well with the addition of a required course on machine learning for software engineers. The new course introduces students to important topics in machine learning and data science, therefore leaving room for more advanced technical electives that could introduce topics such as deep learning or reinforcement learning.

Removal of existing content:
The addition of curriculum content also meant that some existing content would need to be removed to balance student workload while considering accreditation requirements. The existing course on signals and systems was deemed more appropriate to electrical engineering students than the software engineering program. While some of the topics may be helpful for signal

processing applications, the course content is not broadly relevant for a general software engineering degree.

Based on the related subcommittee feedback, the course on software requirements was also converted to an elective rather than a required course. An introduction to software requirements is covered within the newer course content, allowing the existing course to become a more advanced technical elective for students who want a more in-depth approach to requirements engineering. After discussion, faculty members felt that not all students would need the rigour introduced in this course and that students should be introduced to requirements interpretation earlier in their studies.

Suggestions for delivery:
Several opportunities for cross-course collaborations and projects were identified within the curriculum. For example, the winter semester of the second year involves Java programming and concepts across courses on full-stack development, object-oriented programming, and data structures concepts. In the past curriculum, students found the differences between the JDK versions and course requirements to be confusing across similar course objectives. The content can be better aligned to provide one consistent Java setup at the beginning of the semester, and to align touchpoints on the creation/subsequent use of data structures throughout the semester. Recommendations were made for the instructors to function as a cohort team to allow similar expectations and specifications across the courses. A combined course project was piloted in similar graduate level courses where the front-end application is development in one course and the back-end database design in another [3]. Students only enrolled in one of the two courses would not be disadvantaged as the two components can be developed and run independently. This pilot was used to create recommendations for the second-year cohort.

The hardware subcommittee also provided recommendations software-specific content that may be more applicable to software engineering students. Courses that used to be offered across both electrical and software engineering were split into separate offerings that allow the instructor to better strengthen the applicability of the content and its relation to both prerequisite and subsequent knowledge. For example, previous software engineering students had reported struggling with the embedded systems content. While it is beneficial for students to learn about embedded systems programming, there is a need to tie together programming and the control of embedded peripherals while explaining potential ties to applications such as robotics.

Continual Improvement and Feedback

April 2024 will mark the end of the initial curriculum rollout and the first implementation of all changes. An initial survey was sent in Fall 2023 to students in their current second year and beyond as an initial assessment of academic experience and factors included course content, workload, stress, engineering identity, graduate attributes, and more. This survey will be repeated in the summer of 2024 to determine continual improvement measures and to assess any additional modifications that may be needed within the curriculum redesign. An initial review of the survey feedback (n=74) showed that students who went through the new curriculum are more likely to agree or strongly agree that the course content has been valuable and relevant to their future careers. Table 1 shows a comparison between students who began their software

engineering studies prior to Fall 2022, and those who began after Fall 2022 with the new curriculum.

Table 1: Students who responded “agree” or “strongly agree” when asked “Thinking back to your 2nd year courses, please rate the extent to which you agree or disagree with the following statements.”

Statement	**Previous Curriculum Cohort**	**Updated Curriculum Cohort**
I enjoyed my experience	50.00%	63.33%
I often felt overwhelmed by the amount of work expected of me	72.22%	68.97%
I gained valuable knowledge of fundamental engineering principles	66.67%	96.55%
I gained skills that have been useful in my present endeavors	66.67%	93.10%
Practical relationships between different courses/subjects were evident	63.16%	82.76%
The courses I took prepared me for my future as an engineer	50.00%	83.33%
I felt that a community of learning existed	33.33%	63.33%

Future Steps

While the initial feedback of the new curriculum is promising, there are still improvements that need to be made. The rapid new hiring of faculty members has resulted in the need for more familiarity across the curriculum map as well as building collaborations for cross-course integrations. Next steps also include the expansion of technical elective offerings that advance the concepts introduced in the mandatory courses. The development of a continual improvement process is on-going and will continue to be shared with the software engineering education community.

References

[1] B. Tenbergen, S. Krusche, R. Hanna, M. Bano, “Software Engineering Education and Training: Industry Demands, Curriculum Deficits, and Pedagogy Direction”, *Focus: Guest Editors’ Introduction in IEEE Software, IEEE Computer Society*, Nov/Dec 2023, pp. 36 – 39. DOI: 10.1109/MS.2023.3328465

[2] K. A. Gary, R. Acuna, A. Mehlhase, R. Heinrichs, S. Sohoni, “Scaling to Meet the Online Demand in Software Engineering”, *International Journal on Innovations in Online Education*, vol. 4(1), pp. 1-26, 2020.

[3] Canadian Engineering Accreditation Board, “Accreditation Criteria and Procedures”, 2021.

[4] A. Kolmos, R. G. Hadgraft, J. E. Holgaard, “Response strategies for curriculum change in engineering,” *International Journal of Technology and Design Education*, vol. 26, pp. 391-441, 2016. DOI: 10.1007/s10798-015-9319-y

[5] Y. Afshar, M. Moshirpour, E. Marasco, J. Kawash, L. Behjat, and M. Moussavi, “An Integrated Software Engineering Curriculum Through Project-Based Learning (PBL),” in *Proceedings of the American Society for Engineering Education, ASEE 2022 Annual Conference, Minneapolis, Minnesota, USA*.