

**Proposal for a New Academic Program**

**Institution:** Oregon State University - Cascades

**College/School:** College of Engineering

**Department/Program Name:** Electrical Engineering & Computer Science

**Degree and Program Title:** Bachelor of Science, Software Engineering

1. **Program Description**
2. Proposed Classification of Instructional Programs (CIP) number.

14.0903, Computer Software Engineering

1. Brief overview (1-2 paragraphs) of the proposed program, including its disciplinary foundations and connections; program objectives; programmatic focus; degree, certificate, minor, and concentrations offered.

People tend to equate computer science with “learning to code” or computer programming: the practice of writing software that causes computers to do something, ideally solving meaningful problems in a variety of domains. Indeed, students wishing to learn how to create software arrive at an undergraduate institution to find that their intellectual interest fits closest with computer science, often the only major with “computer” in its title. But the academic discipline of computer science is not about writing software.

Computer science (CS) is founded upon mathematics, and is a study of automating algorithms, expanding theories of computation, and designing computational systems that can execute algorithms at scale. A computer scientist is a specialist in the theory of computation and the design of computers. Computer engineering (CE), also offered at many institutions, is an integration of electrical engineering and computer science, with a focus on designing computer hardware, such as circuit design and systems integration. Management information systems (MIS), often offered by business schools, is the study of people, technology, teamwork, leadership, project management, customer service, organizations, and business theories. MIS professionals design and implement information systems to meet the needs of an organization. The world needs computer scientists, computer engineers, and MIS professionals. But more than ever, the world needs people who can build, with code, software solutions to important problems across numerous domains.

Software engineering is the development (coding/programming), operation and maintenance of software, and the study of technical knowledge, methods, and experience to the design, testing and documentation of software (IEEE, 2010). It is an engineering discipline concerned with all aspects of software production. The classic definition, from the birth of software engineering in 1972, is “the establishment and use of sound engineering principles in order to economically obtain software that is reliable and works efficiently on real machines.” (Bauer, 1972). For the sake of this proposal, we define software engineering simply as *the multi-disciplinary study and practice of the technologies, principles, and methodologies in designing, testing, building and maintaining software*. There is demand for knowledgeable software engineers in both industry and research (XXX CITE), with meaningful problems in numerous domains. As an academic pursuit, software engineering has grown into a body of knowledge that stands on its own (XXX CITE). Yet what is higher education doing to educate the next generation of software engineers? The answer is, not much. Because computer science doesn’t teach software engineering.. it teaches the science of computing.

Over the past decade, the market has responded to the lack of available software engineering education with “code schools” and “boot camps.” These programs offer skills-centered training on specific technology stacks (tools and programming languages) to prepare the student for an entry-level software developer position. Despite their shortcomings, such programs have been immensely successful, changing students’ lives, catalyzing economic growth and educating students about the fundamentals of real-world computer programming.

The problem with code schools is that they cannot cheat time. Some of the shortcomings of code school graduates include 1) only having surface knowledge of popular tools and technologies, 2) a lack of strong verbal and written communication skills, 3) a lack of strong critical thinking skills, 4) being trained for a job, rather than having dedication to lifelong learning (Economist, 2017), and 5) a lack of understanding of computer science fundamentals. This academic program proposal avoids these shortcomings with a four-year program in software engineering that synthesizes a liberal arts education with the principles of software engineering and the fundamentals of computer science.

1. Course of study – proposed curriculum, including course numbers, titles, and credit hours.

The software engineering program differs from computer science in that it emphasizes the communication skills, methodologies, tools, technologies, teamwork, professional practice, design, and architectures critical for building scalable, long-lasting software systems. The program emphasizes these software engineering principles over, but not at the exclusion of, computer science principles. The curriculum includes high-impact, team- and project-based courses early in the curriculum, followed by computer science courses that explore the most important theoretical topics in computer science.

During year one, students engage in a team-based, real-world project in which they create a “full stack” application covering the significant topics in software engineering principles and industrial practice. While learning outcomes are not at the mastery level, students repeat this project-based experience going both deeper into all topics as well as focusing on the second year theme of “data”; that is, database management, statistics, data science, data mining, and data visualization.

Year three emphasizes the themes of scalability, security and operations and enables students to continue to move toward mastering the software engineering body of knowledge. In addition, year three marks a transition: the experience of computer programming and software engineering serves as a foundation for exploring deeper computer science topics. Year four emphasizes entrepreneurship and the business of software, complemented by additional computer science courses.

Lastly, throughout all four years of the curriculum, the software engineering program integrates experiential learning and professional practice. The program expects that each student be engaged in a professional apprenticeship throughout their academic career.

The program of study follows the degree standards at Oregon State University, incorporating both existing courses and new courses. The “SE” prefix indicates new courses, which are also followed by the CAT II proposal numbers (99XXX).

**\*Freshman Skill Courses (16 credits) #BACC Core (48 cr) 99XXX- CAT II course proposal**

|  |
| --- |
| First Year 46 cr |
| Fall **15 cr**  SE 101: Creative Problem Solving with Code I (2) (99XXX)  SE 111: Introduction to Software Engineering I (6) (99XXX)  SE 107: Professional Seminar (1) (99XXX)  WR 121#\*: English Composition (3)  HHS 231#\*: Lifetime Fitness for Health (2)  PAC XXX#\*: Various Physical Activity Courses (1) |
| Winter **16 cr**  SE 102: Creative Problem Solving with Code II (2) (99XXX)  SE 112: Introduction to Software Engineering II (6) (99XXX)  SE 110: Apprenticeship I (1) (99XXX)  MTH 112#: Elementary Functions (4)  CS 391#: Social and Ethical Issues in Computer Science (3) (Bacc Core Synthesis: Science, Technology, Society) |
| Spring **15 cr**  SE 103: Creative Problem Solving with Code III (2) (99XXX)  SE 113: Introduction to Software Engineering III (6) (99XXX)  SE 110: Apprenticeship I (1) (99XXX)  COMM 111#\*: Public Speaking (3)  WR 327#\*: Technical Writing (3) |
| Second Year 45 cr |
| Fall **15 cr**  SE 211: Data Science Engineering I (6) (99XXX)  SE 210: Apprenticeship II (1) (99XXX)  ST 351: Intro to Statistical Methods (4)  BACC Core#: Biology with Lab (4) |
| Winter **15 cr**  SE 212: Data Science Engineering II (6) (99XXX)  SE 210: Apprenticeship II (1) (99XXX)  ST 352: Intro to Statistical Methods (4)  BACC Core#: Physical Science with Lab (4) |
| Spring **15 cr**  SE 213: Data Science Engineering III (6) (99XXX)  SE 210: Apprenticeship II (1) (99XXX)  MTH 231: Elements of Discrete Mathematics (4)  BACC Core#: Bio or Physical Science with Lab (4) |
| Third Year 48 – 51 cr |
| Fall **16 - 17 cr**  SE 301: Elements of Computing Systems I (2) (99XXX)  SE 311: Scalability, Infrastructure and Security I (6) (99XXX)  SE 310: Apprenticeship III (1) (99XXX)  CS 261: Data Structures (4)  BACC Core#: Western Culture or American History (3 - 4) |
| Winter **16 - 17 cr**  SE 302: Elements of Computing Systems II (2) (99XXX)  SE 312: Scalability, Infrastructure and Security II (6) (99XXX)  SE 310: Apprenticeship III (1) (99XXX)  CS 271: Computer Architecture & Assembly Language (4)  BACC Core#: Cultural Diversity (3 - 4) |
| Spring **16 - 17 cr**  SE 303: Elements of Computing Systems III (2) (99XXX)  SE 313: Scalability, Infrastructure and Security III (6) (99XXX)  SE 310: Apprenticeship III (1) (99XXX)  CS 381: Programming Language Fundamentals (4)  BACC Core#: Literature & Arts (3-4) |
| Fourth Year 36 – 39 cr |
| Fall **12 - 13 cr**  SE 411: Business of Software I (4, WIC) (99XXX)  SE 410: Apprenticeship IV (1) (99XXX)  CS 325: Analysis of Algorithms (4)  BACC Core#: Difference, Power & Discrimination (3 - 4) |
| Winter **12 - 13 cr**  SE 412: Business of Software II (4, WIC) (99XXX)  SE 410: Apprenticeship IV (1) (99XXX)  CS 344: Operating Systems I (4)  BACC Core#: Synthesis, Contemporary Global Issues (3 - 4) |
| Spring **12 - 13 cr**  SE 413: Business of Software III (4, WIC) (99XXX)  SE 410: Apprenticeship IV (1) (99XXX)  CS XXX: Upper-Division CS Elective (4)  BACC Core#: Social Processes & Institutions (3 - 4) |
| **Total Credit Hours 175 - 181** |

**\*Freshman Skill Courses (16 credits) #BACC Core (48 cr) 99XXX- CAT II course proposal**

**OSU Graduation Requirements**

180 total credits: 175 – 181 credits (139 major and 36 - 42 bacc core)

60 upper division credits: 68 upper division credits in major

**OSU Engineering Professional School**

The first two years of the program include “pre-professional” courses unique to the program that must be taken at OSU-Cascades. The second two years are “professional” courses that require admission to the College of Engineering professional school. Grade point average in select pre-professional courses dictates admission. The pre-professional school courses used for admittance into the professional school will follow the same model as those used by many of the other academic programs in the College of Engineering.

1. Manner in which the program will be delivered, including program location (if offered outside of the main campus), course scheduling, and the use of technology (for both on-campus and off-campus delivery).

The OSU-Cascades campus will deliver the Software Engineering program as a face-to-face, on-campus program in Bend, while housed within the OSU College of Engineering. The proposed curriculum schedules all existing (non-SE courses) according to the existing offerings at OSU-Cascades. Courses during the first two years of the curriculum may be scheduled and taken at two-year institutions, such as Central Oregon Community College in Bend, for transfer during a student’s pre-professional enrollment in the program. The program expects each student to own a laptop, and the program will leverage affordable third party services, such as cloud-based hosting, and open source tools.

1. Adequacy and quality of faculty delivering the program.

OSU-Cascades currently offers a computer science program employing two full-time faculty and three to four adjunct instructors. OSU-Cascades plans to sunset computer science as the software engineering program starts, by ceasing enrollment in computer science and opening enrollment in software engineering. In addition, OSU-Cascades will hire one additional full-time faculty member, specializing in the field of software engineering, before the first year of the software engineering program. During the transition, these three full-time faculty will teach part of the remaining computer science courses and the new software engineering courses. The faculty include:

* Yong Bakos, MS Software Engineering, Instructor
  + With over twenty years of experience as a software engineer, and over ten years in computer science education, Bakos will lead the transition to the software engineering curriculum.
  + Teaching SE 107, 111, 112, 113, 211, 212, 213, 301, 302, 303
* Dr. Marc Rubin, PhD Computer Science, Instructor
  + Currently the computer science program lead at OSU-Cascades, Rubin will lead the delivery of the computer science courses in the new software engineering curriculum.
  + Teaching SE 101, 102, 103, and CS 261, 271, 381, 325, 344, CS elective
* Third full-time faculty member, PhD, Assistant Professor
  + An upcoming hire, this role represents a tenured or tenure-track position with research in the field of software engineering or computer science education.
  + Teaching SE 311, 312, 313, 110, 210, 310, 410
* One to two part-time instructors, software professionals in the field
  + OSU-Cascades has successfully employed software professionals employed in Bend, specializing in software engineering, usability, networking, security, web and mobile software development, and computer science topics.
  + Teaching SE 411, 412, 413, and CS 391

1. Adequacy of faculty resources – full-time, part-time, adjunct.

Beyond the faculty listed in section 1e, above, OSU-Cascades envisions hiring up to two additional full-time faculty and/or multiple part-time faculty to teach additional sections of courses. Based on prior experience, the Bend business environment offers a strong and growing pool of part-time faculty with real-world experience in software engineering.

1. Other staff.

Current OSU-Cascades staff will manage recruitment, enrollment, and advising until program growth dictates the need for additional staff.

1. Adequacy of facilities, library, and other resources.

The two current full-time faculty currently occupy individual offices, and OSU-Cascades has an additional office for the third full-time faculty. Part-time faculty do not require dedicated office space.

Existing classroom facilities are adequate and already exist to support the software engineering program. The program ideally utilizes the “Learning Studio” on the OSU-Cascades campus,



but any existing classrooms are sufficient. With the planned expansion of the OSU-Cascades campus, sufficient classroom and office space will be available (see letter of evaluation).

The existing library resources were judged as satisfactory (see letter of evaluation).

TODO: letter from Laurel Kristick

TODO: letter from Jane Barker

1. Anticipated start date.

The software engineering program begins Fall 2019.

We anticipate Winter 2018 for program approval, and we will begin recruitment of our first cohort of students at that time. In Fall 2019, we will offer our undergraduate pre-professional coursework. Winter 2020 will be our first round of admissions for the junior cohort with professional coursework to start in Fall 2021.

1. **Relationship to Mission and Goals**
2. Manner in which the proposed program supports the institution’s mission, signature areas of focus, and strategic priorities.

*As a land grant institution committed to teaching, research and outreach and engagement, Oregon State University promotes economic, social, cultural and environmental progress for the people of Oregon, the nation and the world* (OSU, 2017).

The Software Engineering program incorporates effective, research-supported pedagogy within a hands-on classroom and curriculum design, organizes opportunities for student and faculty outreach into the community of Central Oregon, and integrates economic, social, cultural and environmental issues through world-relevant coursework and professional practice.

*This mission is achieved by producing graduates competitive in the global economy, supporting a continuous search for new knowledge and solutions and maintaining a rigorous focus on academic excellence… (OSU, 2017)*

The Software Engineering program aspires to become a world-class educational experience that transforms its students into emerging leaders who can design and deliver large, scalable, high-performing, and maintainable software systems, (an area of high demand across the global economy). The cross-disciplinary, project-based curriculum design, integration with professional practice, synthesis of computer science topics, and incorporation of continuously evolving technologies supports the achievement of the OSU mission.

*…particularly in the three Signature Areas: Advancing the Science of Sustainable Earth Ecosystems, Improving Human Health and Wellness, and Promoting Economic Growth and Social Progress (OSU, 2017).*

The Software Engineering program is inherently cross-disciplinary, and positions itself across OSU’s three Signature Areas. For example, students may engineer a software system that transmits, aggregates, mines and visualizes sensor data for river restoration; they may engage with a local medical device company to improve software reliability and security, or build applications that assist the disabled; and, they may integrate new systems, techniques and solutions that they bring to market through their own startup business, creating new jobs and economic growth. advancing science of sustainable.. by …”

Strategically, one priority for the OSU-Cascades campus is to distinguish itself with innovative, world-class, “destination degree programs” not offered at most institutions. The Software Engineering program exemplifies these qualities, following the success of the existing Energy Systems Engineering program, and co-existing with other innovative new programs, such as Outdoor Products.

1. Manner in which the proposed program contributes to institutional and statewide goals for student access and diversity, quality learning, research, knowledge creation and innovation, and economic and cultural support of Oregon and its communities.

The Software Engineering program approaches student access and diversity as a continuum, beginning with K-12 outreach (Bush & Miller, 2017; Wilson et al, 2010) and bringing K-12 students onto campus to observe Software Engineering classes and students (Engle et al, 2006). The program ensures an approachable introductory course sequence for students with diverse backgrounds (Cohoon & Tychonievich, 2011), implements an authentic, hands-on, and world-relevant active learning experience (Eisenberg, 2017), cultivates a community of supportive peers through specific learning methods (Nagappan, 2013), and integrates professional exposure early in the curriculum, to support student career-readiness upon graduation.

The Software Engineering program has, as its advantage, the ability to implement effective teaching and learning techniques, and experiences for knowledge creation, based on decades of prior research in engineering and computer science education. An advantage of the proposed SE program is its ability to implement proven pedagogical methods, based on decades of prior research, directly into the curriculum Furthermore, the engagement of real-world problems, elicited from local, regional and statewide businesses directly contributes to innovation, economic and cultural support of Oregon.

1. Manner in which the program meets regional or statewide needs and enhances the state’s capacity to:
2. improve educational attainment in the region and state;
3. respond effectively to social, economic, and environmental challenges and opportunities; and
4. address civic and cultural demands of citizenship.

Central Oregon is an education desert or a “community where students have few postsecondary options from which they can choose.” Nationally, students travel less than 50 miles to attend a 4-year public university. OSU-Cascades is a critical player in the social and economic development of Central Oregon and cannot serve the needs of the students and employers of the region without further expansion of academic programs and degrees. In addition, Central Oregon Community College’s role cannot be understated in Central Oregon, where over 86% of all students in higher education are enrolled. The Software Engineering program capitalizes on the lower division preparation provided by COCC to remote communities to recruit diverse students. By working in partnership with COCC, we have created a pathway to the undergraduate degree that can be started at either institution.



“The zip code that a child is born into oftentimes determines their life chances.” (Hillman & Weichman, 2016)

1. **Accreditation**
2. Accrediting body or professional society that has established standards in the area in which the program lies, if applicable.

The Accreditation Board for Engineering and Technology (ABET) provides accreditation to engineering programs in the United States. The Computing Sciences Accreditation Board (CSAB) leads the ABET Engineering Accreditation Commission. Members include the Association of Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers (IEEE).

1. Ability of the program to meet professional accreditation standards. If the program does not or cannot meet those standards, the proposal should identify the area(s) in which it is deficient and indicate steps needed to qualify the program for accreditation and date by which it would be expected to be fully accredited.

The Software Engineering program outcomes and quality assessment (Section 5) specifically include the outcomes and standards set forth by the Criteria for Accrediting Engineering Programs (ABET, 2017b). The courses and course learning outcomes cover the “breadth and depth of engineering and computer science topics” in the Program Criteria for Software and Similarly Named Engineering Programs (ABET, 2017b). Identical to the accreditation criteria, the Software Engineering curriculum includes computing fundamentals, software design and construction, requirements analysis, security, verification, and validation; software engineering processes and tools appropriate for the development of complex software systems; and discrete mathematics, probability, and statistics, with applications appropriate to software engineering.

1. If the proposed program is a graduate program in which the institution offers an undergraduate program, proposal should identify whether or not the undergraduate program is accredited and, if not, what would be required to qualify it for accreditation.

N/A

1. If accreditation is a goal, the proposal should identify the steps being taken to achieve accreditation. If the program is not seeking accreditation, the proposal should indicate why it is not.

Following ABET guidelines, we will record data for the required Self-Study Report during the first three years of the Software Engineering program. We will submit a Request for Evaluation in the third year of the program, and plan to obtain accreditation by the program’s fourth or fifth year of operation (ABET, 2017a).

1. **Need**
2. Anticipated fall term headcount and FTE enrollment over each of the next five years.
3. Expected degrees/certificates produced over the next five years.
4. Characteristics of students to be served (resident/nonresident/international; traditional/ nontraditional; full-time/part-time, etc.).
5. Evidence of market demand.
6. If the program’s location is shared with another similar Oregon public university program, the proposal should provide externally validated evidence of need (e.g., surveys, focus groups, documented requests, occupational/employment statistics and forecasts).
7. Estimate the prospects for success of program graduates (employment or graduate school) and consideration of licensure, if appropriate. What are the expected career paths for students in this program?
8. **Outcomes and Quality Assessment**
9. Expected learning outcomes of the program.
10. Methods by which the learning outcomes will be assessed and used to improve curriculum and instruction.
11. Nature and level of research and/or scholarly work expected of program faculty; indicators of success in those areas.
12. **Program Integration and Collaboration**
13. Closely related programs in this or other Oregon colleges and universities.
14. Ways in which the program complements other similar programs in other Oregon institutions and other related programs at this institution. Proposal should identify the potential for collaboration.
15. If applicable, proposal should state why this program may not be collaborating with existing similar programs.
16. Potential impacts on other programs.

**7. External Review**

If the proposed program is a graduate level program, follow the guidelines provided in *External Review of New Graduate Level Academic Programs* in addition to completing all of the above information.

*Revised May 2016*

**References**

ABET (2017a). Accreditation Policy and Procedure Manual (APPM), 2017 – 2018. Retrieved from<http://www.abet.org/accreditation/accreditation-criteria/accreditation-policy-and-procedure-manual-appm-2017-2018/>

ABET (2017b). Criteria for Accrediting Engineering Programs, 2017 – 2018. Retrieved from<http://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-engineering-programs-2017-2018/>

Bush, J. & Miller, S. (2017). Analysis of Associations between Motivation and Previous Computer Science Experience, Gender, Ethnicity and Privilege as Observed in a Large Scale Survey of Middle School Students. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (SIGCSE '17). ACM, New York, NY, USA, 705-705. doi:<https://doi.org/10.1145/3017680.3022441>

Cohoon, J. & Tychonievich, L. (2011). Analysis of a CS1 approach for attracting diverse and inexperienced students to computing majors. In *Proceedings of the 42nd ACM technical symposium on Computer science education* (SIGCSE '11). ACM, New York, NY, USA, 165-170. doi:<http://dx.doi.org/10.1145/1953163.1953217>

Eisenberg M. (2017) Approaching Computer Science Education Through Making. In: Fee S., Holland-Minkley A., Lombardi T. (eds) *New Directions for Computing Education*. Springer, Cham. doi:<https://doi.org/10.1007/978-3-319-54226-3_3>

Engle, J., Bermeo, A., O’Brien, C. (2006). Straight from the Source: What Works for First-Generation College Students. Washington, DC: Pell Institute for the Study of Opportunity in Higher Education. Retrieved from<http://www.pellinstitute.org/downloads/publications-Straight_from_the_Source.pdf>

Hillman, N. & Weichman, T. (2016). Education Deserts: The Continued Significance of “Place” in the Twenty-First Century. *Viewpoints: Voices from the Field*. Washington, DC: American Council on Education. Retrieved from<http://www.acenet.edu/news-room/Documents/Education-Deserts-The-Continued-Significance-of-Place-in-the-Twenty-First-Century.pdf>

Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C., & Balik, S. (2003). Improving the CS1 experience with pair programming. In *Proceedings of the 34th SIGCSE technical symposium on Computer science education* (SIGCSE '03). ACM, New York, NY, USA, 359-362. DOI=http://dx.doi.org/10.1145/611892.612006

Oregon State University (2017). Oregon State University Mission Statement. Retrieved from<http://leadership.oregonstate.edu/trustees/oregon-state-university-mission-statement>

Wilson, C., Sudol, L. A., Stephenson, C. & Stehlik, M. 2010. Running on Empty: The Failure to Teach K-12 Computer Science in the Digital Age. The Association for Computing Machinery & The Computer Science Teachers Association. Retrieved from http://runningonempty.acm.org/fullreport2.pdf