BS Computer Science

Applied Option, Software Engineering

The following proposed Applied Option in Software Engineering supplements the Computer Science core requirements with courses that:

1. Bridge the gap between introductory programming (1xx) and software engineering (3xx)
2. Address modern industry needs, (e.g. recent IAB feedback)
3. Provide an intermediate step toward a four-year BS in Software Engineering

The Applied Option in Software Engineering includes existing 400-level CS electives and four new courses:

* One 200-level course on the development of larger software applications
* One 300-level course that extends the existing CS 361 & 362 sequence
* Two 400-level courses that extend the existing CS 466 course

These courses provide students with additional software development practice during year two, extend the existing software engineering course sequence in CS during year three, and extend the existing startup/entrepreneurship course in CS in year four, complementing the CS capstone sequence.

This applied option requires four Category 2 proposals for new courses, and the development of these four courses. Cascades campus faculty shall complete this work during AY2018-19, and make the applied option available in AY2019-20.

SE 201 Software Development I

**Rationale**

Students experience programming “in the small” in year one (CS 160-162), experience very few courses in year two (CS 261, 271 and 290), and then engage in “software engineering” (CS 361 & 362) in year three. Learning software engineering concepts can be challenging when students have only written short, focused programs and have never engaged in building a larger software system. The SE 201 course engages students in programming “in the large” (more OO, more practice, more APIs, more problem solving, more tools, bigger programs) in year two, providing a learning experience that helps bridge the first year and third year courses. *In the four-year BS Software Engineering curriculum, this course will be the first in a three-course sequence (SE 201 – 203).*

**Catalog Description**

Introduction to collaborative software development of larger, object-oriented systems. Overview of software architecture, and the tools, principles and practice of modern software development.

**Learning Outcomes**

1. Create software application solutions that solve customer-defined problems and meet user requirements.
2. Analyze a problem domain to create appropriate abstractions.
3. Apply software engineering design principles and practice writing high-quality code.
4. Assess the maintainability and quality of code, and apply basic refactoring techniques.
5. Practice using tools to facilitate the implementation of software applications.
6. Discover and explain the features of third-party libraries and services; and integrate third-party APIs into a software system.
7. Work in a team to design a moderately-sized software system.
8. Define the terminology and describe the basic concepts of software architecture.

SE 303 Software Engineering III

**Rationale**

The CS 361 & 362 courses address the “front end” and “back end” of a software lifecycle, from requirements analysis to verification and maintenance. Modern software engineering is a huge topic, and even our IAB has raised some curricular shortcomings, including continuous integration and delivery, automated testing, cloud computing, and infrastructure. A third course in the software engineering sequence provides more surface area for such topics and additional learning outcomes. *In the four-year BS Software Engineering curriculum, this course will be the third of a three-course sequence (CS 361, CS 362, SE 303).*

**Catalog Description**

Introduction to refactoring techniques and improving the quality and maintainability of software. Applying continuous integration and deployment tools; containers and virtual development environments.

**Learning Outcomes**

1. Recognize flaws in software design (e.g. “code smells”).
2. Describe common refactoring techniques.
3. Apply refactoring techniques in a test-driven process to improve the quality and maintainability of code.
4. Describe deployment processes such as continuous integration and continuous delivery.
5. Apply tools for managing the construction and deployment of N-tier applications for virtualized environments.
6. Describe the benefits and drawbacks of containers for managing development and production environments.
7. Apply containerization tools to create development and production environments.
8. Participate effectively in a software engineering team.

SE 402 & 403 Business of Software II & III

**Rationale**

The capstone sequence (CS 461-463) provides students a design experience in which they typically engage in a consulting model providing a service to a client. In contrast, a product-based model (such as a software startup) provides students an entrepreneurial experience that includes market analysis, product management, design, marketing, business organization, business administration, selling software, and growing a business. While the CS 466 course introduces students to entrepreneurship fundamentals, it does not provide students the time and depth necessary for a true startup experience. SE 402 and 403 shall follow CS 466 to extend this experience, and complement the capstone experience. *In the four-year BS Software Engineering curriculum, these courses will be the second and third of a three-course sequence (CS 466, SE 402, SE 403).*

**Catalog Description**

TODO

466:

Real-world, hands-on learning in a high-tech web/mobile-based company environment. Research in the development of product ideas, hypotheses, and business models to create customer experiments. Prototyping and statistical analysis to develop, optimize, and evaluate solutions. Rapid iteration/refactoring based on customer input, web analytics, and user engagement metrics.

461-3:

Utilize software engineering methodology in a team environment to develop a real-world application. Teams will be responsible for all phases of software development, including project planning, requirements analysis, design, coding, testing, configuration management, quality assurance, documentation, and delivery. Three-term sequence required.

**Learning Outcomes**

1. Execute a software-as-a-service (SaaS) business model, from ideation to sales.
2. Implement product management processes and techniques to define, iterate and scale a software product.
3. Present a business and its product to potential investors.
4. Design, build, and scale a software-as-a-service platform using software engineering techniques.
5. Integrate e-commerce tools and systems for monetary transactions.
6. Execute a marketing and sales plan to generate customer sales.
7. Manage a startup business.
8. Improve a software product using iterative software engineering techniques to meet customer requirements and market demand.
9. Participate effectively in a team environment.

466:

* Identify the key elements of a business model and explain the importance of articulating and testing the assumptions related to web and mobile web startups
* Interview customers to iterate and refine key assumptions comprising the value proposition and business model
* Identify state-of-the-art technologies available in web/mobile delivery
* Design and deliver a minimal viable product in a web or mobile application
* Synthesize customer feedback to refine product-market fit
* Explain importance of build-measure-learn process

461-3:

1. Design, plan, organize, synthesize and complete a significant software project in three academic quarters

2. Apply all aspects of the software engineering process, including project planning, requirements documents, software design, coding, testing, walk-throughs, documentation and delivery

3. Demonstrate good communication skills in the form of weekly reports and project talks, posters, and elevator talks

4.

6. Evaluate the professional, legal, and/or social implications of software product development

7. Evaluate the contributions and importance of software projects to the broad user community

8. Explain the importance of software projects to people from other disciplines and the general public

**Applied Option Criteria**

“Options consist of a minimum of 21 designated quarter credits of related course work, 15 of which must be at the upper-division level.” (Office of Academic Programs & Assessment)

The Applied CS Program must meet the following conditions:

* Minimum of 32 credits
* Recommended minimum of 20 upper division credits
* Coherent body of knowledge where applications of computer science could play an important role

## Core (20 credits)

SE 201 Software Development I (4)

SE 303 Software Engineering III (4)

CS 466 Web-based Startup Project (4)

SE 402 Business of Software II (4)

SE 403 Business of Software III (4)

## Electives (12 credits)

Choose 12 credits from the following:

CS 312 System Administration (4)

CS 373 Defense Against the Dark Arts (4)

CS 440 Database Management Systems (4)

CS 450 Introduction to Computer Graphics (4)

CS 458 Introduction to Information Visualization (4)

CS 475 Introduction to Parallel Programming (4)

CS 492 Mobile Software Development (4)

CS 493 Cloud Application Development (4)

CS 495 Interactive Multimedia Projects (4)

CS 464 Open Source Software (4)

ECE 478 Network Security (4)

CS 434 Machine Learning & Data Mining (4)

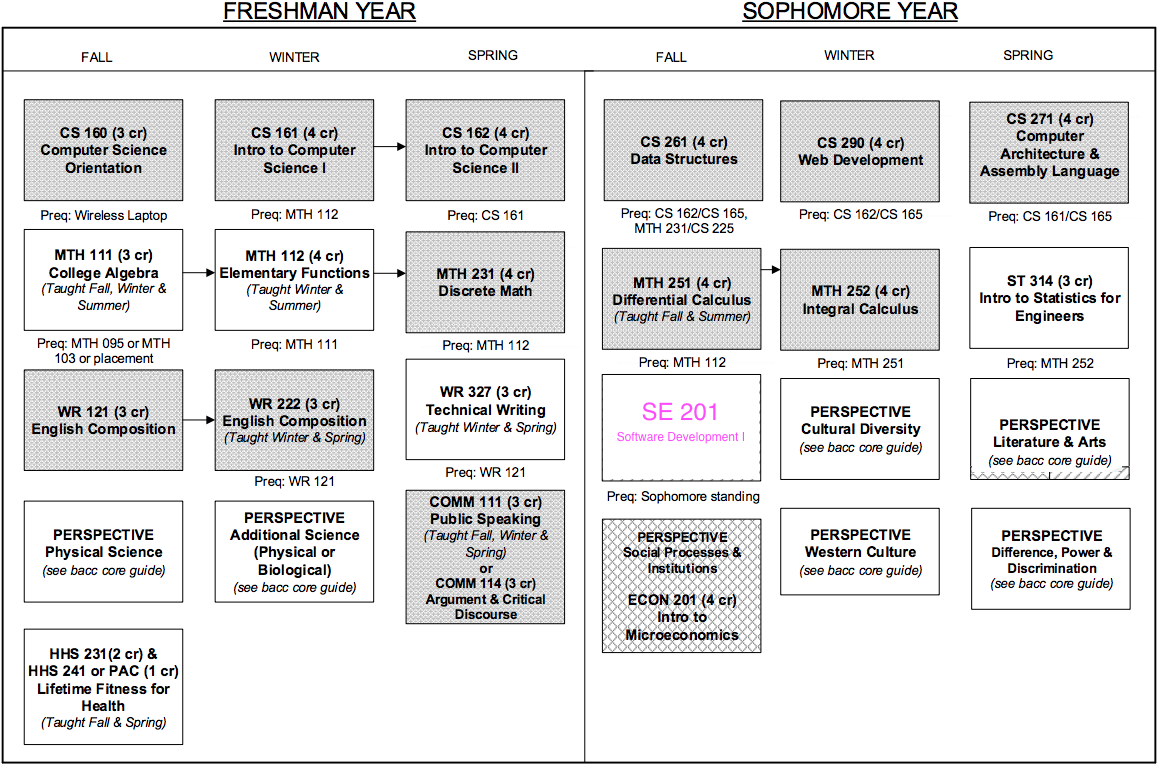
CS 447 Wireless Embedded Systems (4)

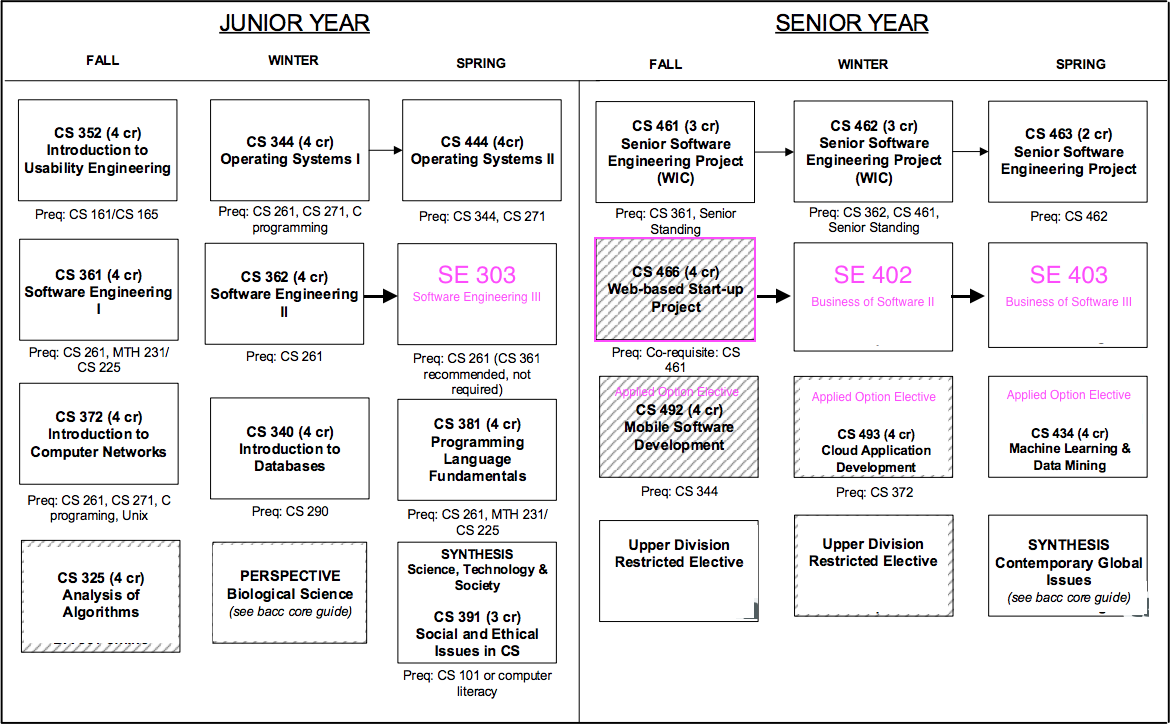
CS 406 Projects (4) or CS 401 Research (4)

4 credits at lower division (SE 201)

28 credits at upper division (SE 303, CS 466, SE 402, SE 403, and CS electives)

Example Four-Year Plan





Current Applied Option, Web and Mobile Software Development



Current Four-Year Plan, Web and Mobile Software Development

