CONTACT INFORMATION

Patrick Conroy, BSc, MSc, MBA, MASc, Eng.L
Department of Electrical and Computer Engineering
Faculty of Applied Science
University of British Columbia
pfconroy@gmail.com

CONTEXT

Role in UBC Programs

This course is restricted to students enrolled in the following programs:
- BASC years 2 or higher in the CPEN specialization
- BASC years 3 or higher in the CPSC specialization
- BCS years 2 or higher

Prerequisites/Co-Requisites

This course has at least one of the following prerequisites: EECE 310, CPEN 310, or CPSC 310.
This course has no co-requisites.

OVERVIEW AND MOTIVATION

The idea of a 'project', i.e. an organized set of activities with a defined set of outcomes achieved with specified resources in a stipulated period of time, is present in virtually every type of human endeavor, and much has been learned about how to plan, execute, monitor, adjust and evaluate projects in general. Attempts have been made to codify and systematize the structure and operation of projects. One effort in particular, the Project Management Institute’s Project Management Book of Knowledge (PMBoK), is an impressive codification largely reflecting historical technical disciplines, notably manufacturing, and the attempts to apply it to software development projects are many and diverse. Another is Carnegie Mellon’s Software Engineering Institute’s Capability Maturity Model (CMM) and its derivatives, aimed at assessing and improving organizational capability to execute software projects. Then there is the large number of execution frameworks and tools that are to the stock-in-trade of software project managers all over the world. This course will expose students to these attempts to codify the software development process and the most ubiquitous of the frameworks and tools for managing that process in action.

However, it has become clear that there are elements of typical software development projects that make them often greatly resistant to the kind of codification represented by the PMBoK, CMM, and the frameworks and tools employed. The evidence for this is a rather dismal record of software quality, many well-documented disasters, as well as frequent attempts at frameworks with the goal of solving these problems. Some of the reasons for such negative outcomes are now obvious, notably that the manufacturing background that was the inspiration for much of the codification and framework/tool effort is incompatible with most software development projects. In particular, software development generally produces singular, highly unique outputs whereas the projects that inspired the PMBoK and the CMM produced large quantities of identical products. Also, software projects are very often exploratory in nature throughout all stages, requiring an iterative try/fix approach fundamentally different than a typically linear design/manufacture.
process. More subtly, software development is human-intensive at every stage, bringing into play what turns out to be a large array of biases and blind spots not present in manufacturing-type operations past the design stage.

This course explicitly stands as a reminder to aspiring Software Project Managers, that despite the highly technical nature of software products, the epistemology that produces them is fundamentally lacking in the effects of the non-technical components inevitably involved. Indeed, this course will expose two fundamental issues that have traditionally received less attention than is deserved in this respect. First, software projects are throughout their lifecycles, ubiquitously influenced by human-factors judgment. Second, there are many lessons to be learned from other disciplines that have strong application to software project management, most notably from theories of human personal and organizational dynamics.

In short, in order to master the craft of Software Project Management, its practitioners must become as skilled at reading and influencing people as at mastering, adapting and applying the huge variety of technical tools they deal with every day.

**LEARNING OBJECTIVES**

Upon completion of this course, it is expected that students will be able to:

- Grasp the essential characteristics of the most common software project codification schemes and the most common frameworks and tools for managing projects.
- Understand that certain theories of human personal and organizational dynamics should inform the form and practice of software project management.
- Select or adapt the (mix of) software methodologies most appropriate to software projects based on the human/organizational environment in which they operate.
- Identify, characterize and manage risks associated with a software development project.
- Estimate the effort and cost of a software development project, and justify that estimate.
- Defend the project against ‘scope creep’, low quality, and excessive Technical Debt.
- Understand the most important legal and ethical issues associated with software projects.
- Develop and maintain a comprehensive software project development plan.
- Reflect on his/her software-oriented practices and biases.

**STRUCTURE AND OPERATION**

This 3-credit course is delivered by an EECE Instructor and two Teaching Assistants working under the Instructor’s guidance. Lectures, Pre-Readings and Tutorials are the primary means of delivering course content to students. Quizzes, graded Assignments, Tutorial attendance, and a UBC-sanctioned Final Examination are the means of assessing students' mastery of the course material and its implications. There is no Mid-Term Examination for this course.

**Lectures and Reading Assignments**

Generally, Lectures are scheduled twice weekly (none on Statutory Holidays and during Reading Week). Consult your UBC CWL account for specific, fixed, days/times/locations.

Lectures are the primary mechanism in this course for the introduction of concepts and ideas, and to a limited extent, the structure and meaning of the processes and algorithms that embody them. Assignments will generally be assigned during Lectures. Support for Assignment completion will be provided to a certain extent in some Tutorials (see below).

There will be approximately 25 Lectures in this course:

- Pre-reading will generally be assigned in advance of each Lecture. Topics will generally be drawn from the named elements below.
• Most Lectures will begin with a short Quiz about previously-assigned Pre-Reading. Quizzes will generally be conducted in the first 5 minutes of a Lecture; students who are late will not be accommodated at other times. Quiz results count toward final course grades.

• Most Lectures will focus on current software project management frameworks and tools, will introduce students to the detailed machinery of some of them, but will not be ‘training sessions’; students are expected to consult textbooks and other aids to gain mastery of such project management machinery. Some support for Assignments exercising the mentioned machinery will be provided in some Tutorials. Frameworks and tools may include:
  o RUP
  o Waterfall Methodology
  o Spiral Methodology
  o Iterative Methodology
  o Extreme Programming
  o Pairwise Programming
  o Agile, in its many variants
  o Use Case Analysis
  o PERT
  o MS Project
  o Trello
  o Estimation – several variants
  o Risk Profiling
  o Technical Debt
  o Other

• Some Lectures will introduce ideas from theories of human and organizational dynamics that bear on software project management practices. Note: it may not be obvious at first glance how these ideas inform current practice; indeed, the goal here is to uncover, through Lectures, Assignments and Tutorials how these ideas might inform better practices. Students are expected to proactively delve deeper into such material. Theories may include:
  o Actor-Network Theory
  o Behavioral Decision Theory
  o Boundary Objects Theory
  o Weber’s Bureaucracy Theory
  o Competitive Theory
  o Complexity Theory
  o Design Thinking Theory
  o Equity Theory
  o General Systems Theory
  o Institutional Theory
  o Prospect Theory
  o Soft Systems Theory
  o Stakeholder Theory
  o Strukturation Theory
  o Task-Technology Fit Theory
  o Team Cohesion Theory
  o Transactive Memory Theory
  o Work Systems Theory
  o …and perhaps others

Students will be expected to participate in Lectures beyond the above-mentioned Quizzes and may be selected from the Class List during Lectures to answer questions based on what has been covered to that point. Lecture content will be made available to students via Piazza after each Lecture, and all non-private discussions with the Instructor will be posted to Piazza.
Assignments

There will be approximately 8 Assignments in 2 categories:

- ‘Technical’ Assignments: Approximately six Assignments will focus on how to use certain of the current software project management frameworks and tools; they should be considered primarily ‘application’ in nature. These will be assigned approximately every other week in the term, each being due within one week of assignment.
- ‘Human Factors’ Assignments: Two Assignments will focus on how some of the above-listed theories inform current frameworks and how they and other theories might provide a context for more effective frameworks; they should be considered primarily ‘research’ in nature. These will be assigned near the middle of the term with due dates approximately 3-4 weeks from assignment.
- There may be schedule overlap on these Assignments.

All Assignments must be submitted via Canvas, a UBC-sanctioned online software tool available to all students registered in the course, by the dates/times stipulated for each Assignment. Details on obtaining a Canvas account and how to use it for this course will be provided at the beginning of the course.

Tutorials

Generally, one Tutorial in each of two streams (students will have been assigned to one stream or the other) will be scheduled per week during this course (none on Statutory Holidays and during Reading Week), on non-Lecture days (see your CWL account for dates/times/locations).

The focus of the Tutorials is to expand on or illustrate Lecture materials. This will take various forms:
- Illustrate important Project Management principles using games. Tutorials including this kind of instruction are mandatory (attendance will be taken). Games utilized in past years include: Pizzeria Operation (kanban-type project management, bottlenecks, waste prevention, etc.), Mission to Mars (critical tasks, dependencies, unexpected delays, etc.), and Hard Choices (risk management, Technical Debt, etc.).
- Assist students in completing the ‘Technical’ Assignments mentioned above by providing detailed instruction relevant to the tools and techniques needed to complete the Assignments. Tutorials dedicated to this purpose are optional regarding attendance.
- Some Tutorials may feature presentations by highly-accomplished individuals from the software industry who will share their experiences of when theory and practice diverge, and the consequences arising from this. The number of these Tutorials will depend on the Instructor’s success in attracting busy people with little time but with a desire to help out. Attendance at these Tutorials is strongly encouraged due to the quality of the individuals involved, but not mandatory.

A Tutorial schedule will be published in the first week of the course. However, students are alerted that the schedule may change based on the availability of the feature presenters mentioned above. As much notice as possible will be provided, but there are no guarantees.

Final Examination

This course requires that all students physically sit a Final Examination at a date/time/location in the period April 1 through April 30. The Final Examination will consist of approximately 20-40 multiple-choice questions, and approximately 3-6 analysis/discussion questions selected from approximately 8-12 options; allow for 2.5 hours to complete the Final Examination.

Based on previous years’ results, the Final Examination should be considered difficult.
Staff/Student Interaction

Most staff/student interaction for this course will be, in addition to face-to-face interaction during Lectures and Tutorials, via Piazza, a UBC-sanctioned online tool available to all students registered in the course. Details on obtaining a Piazza account and how to use it for this course will be provided at the beginning of the course.

In addition, the Instructor will offer ‘office hours’ (may be via Google Hang-Outs or similar mechanism) and will accept email-anytime interaction with students; Teaching Assistant availability of this kind will be announced at the beginning of the course.

ASSESSMENT STRATEGY AND GRADING SCHEME

Final student grades for this course will be based on the following scheme:

- Lecture Quizzes 10%
- ‘Technical’ Assignments 30%
- ‘Human Factors’ Assignments 20%
- Final Examination 40%

Note that to pass this course, students must attain score at least 50% on the Final Exam as well as on the aggregate grade.

RECOMMENDED TEXTS AND BIBLIOGRAPHY

There is no mandatory textbook for this course. However, the following make very useful contributions to the subject of Software Project Management and may be referenced in the course:


A bibliography, which may be referenced for Pre-Readings and Assignments, will be provided via Piazza at the beginning of the course.