CPEN 321: Software Engineering

piazza.com/ubc.ca/winterterm12016/cpen321/home

Staff, About, Format, Advice, Goals

Instructors

- Farshid Agharebparast
- Sathish Gopalakrishnan

Teaching Assistants

- Bader Alahmad
- John Deppe
- Bibek Kaur
- Theresa Mammarella
- Nick Mulvenna

Reach us using Piazza.

CPEN 321 is about *engineering* software systems

What is Software Engineering?

software engineering ≠ programming

Broad Definition

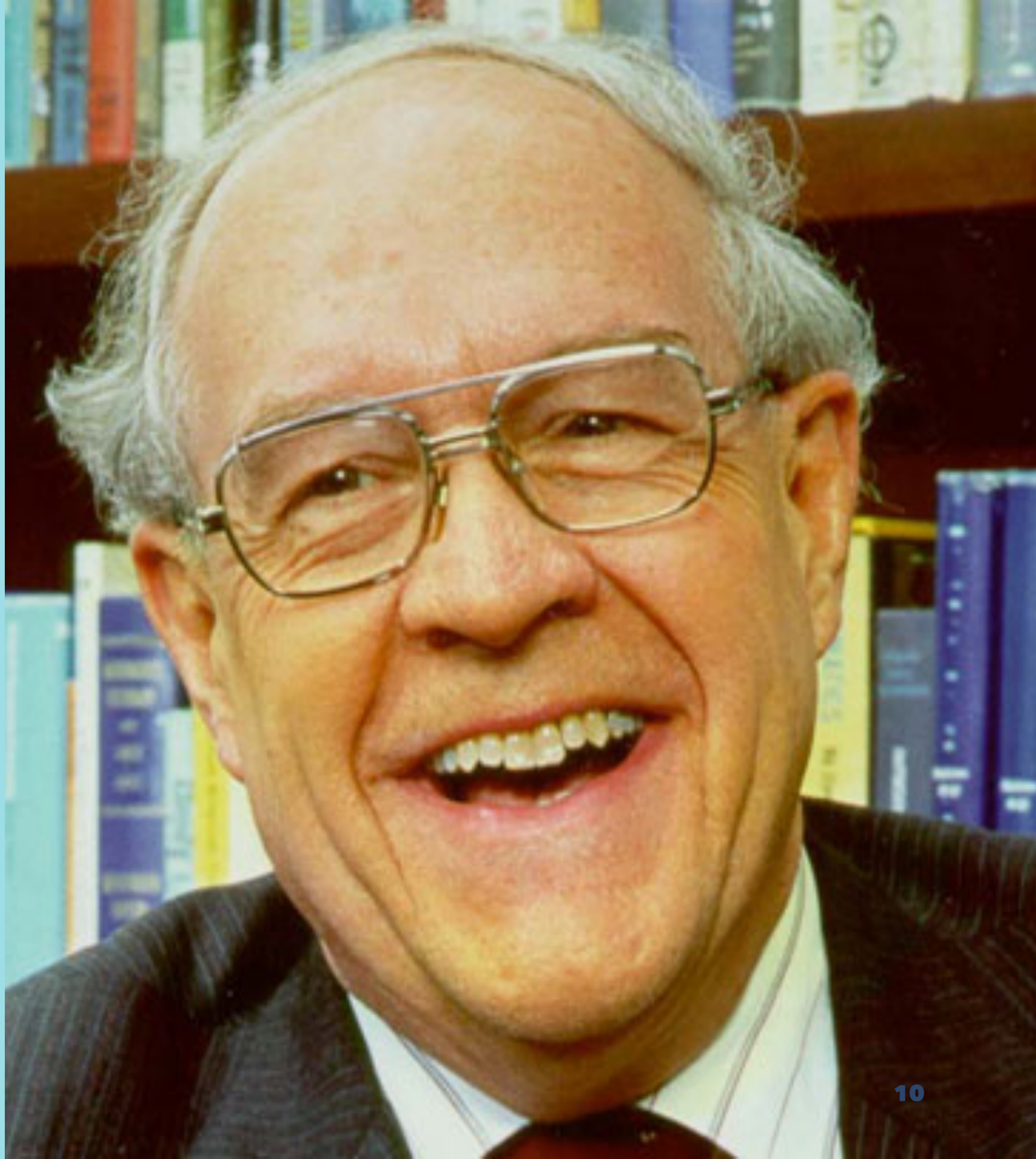
software engineering: creating and maintaining software applications by applying technologies and practices from computing, project management, and other fields.

Software engineering is about people working in teams under constraints to create value for their customers

What is software engineering?

The first step toward the management of disease was replacement of demon theories and humours theories by the germ theory. That very step, the beginning of hope, in itself dashed all hopes of magical solutions. It told workers that progress would be made stepwise, at great effort, and that a persistent, unremitting care would have to be paid to a discipline of cleanliness. So it is with software engineering today.

– Fred Brooks



Some Aspects of Software Engineering

1. Processes,methods,and techniques necessary to turn a concept into a robust deliverable that can evolve over time.
2. Working with limited time and resources.
3. Satisfying a customer.
4. Managing risk.
5. Teamwork and communication.

Software Engineering integrates many disciplines

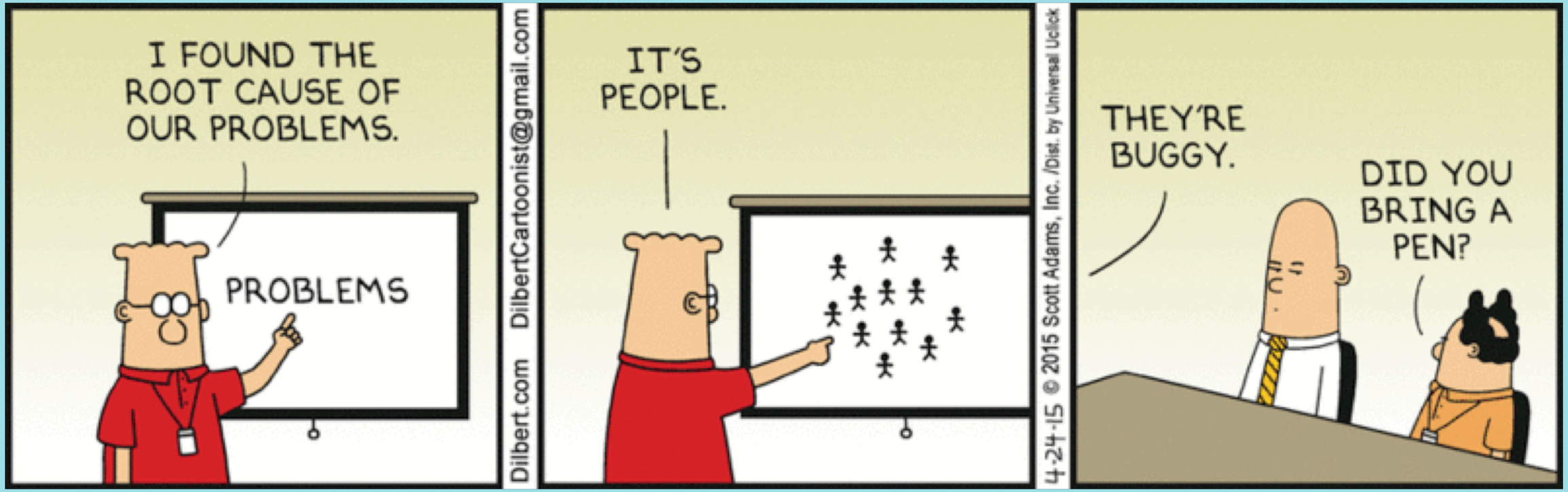
1. computing (programming languages, algorithms, tools)
2. business/management (project management, scheduling)
3. economics/marketing (selling, niche markets, monopolies)
4. communication (managing relations with stakeholders: customers, management, developers, testers, sales)
5. law (patents, licenses, copyrights, reverse engineering) sociology (modern trends in societies, localization, ethics)
6. political science (negotiations; topics at the intersection of law, economics, and global societal trends; public safety)
7. psychology (personalities, styles, usability, what is fun)
8. art (GUI design, what is appealing to users)

Software Engineering is necessarily "softer" than other aspects of computing: fewer right/wrong answers.

"softer" $\neq$ easier

People and Software

- customer / client: wants software built
- managers: make plans, coordinate team
- developers: design and write code
- testers: perform quality assurance (QA)
- users: purchase and use software product


I FOUND THE ROOT CAUSE OF OUR PROBLEMS.
PROBLEMS
Dilbert.com DilbertCartoonist@gmail.com
IT'S PEOPLE.
4-24-15 © 2015 Scott Adams, Inc. /Dist. by Universal Uclick
THEY'RE BUGGY.
DID YOU BRING A PEN?

customer / client: wants software built

often doesn't know what he/she wants

managers: make plans, coordinate team

hard to foresee all problems in advance

developers: design and write code

hard to write complex code for large systems

testers: perform quality assurance (QA)

impossible to test every combination of actions

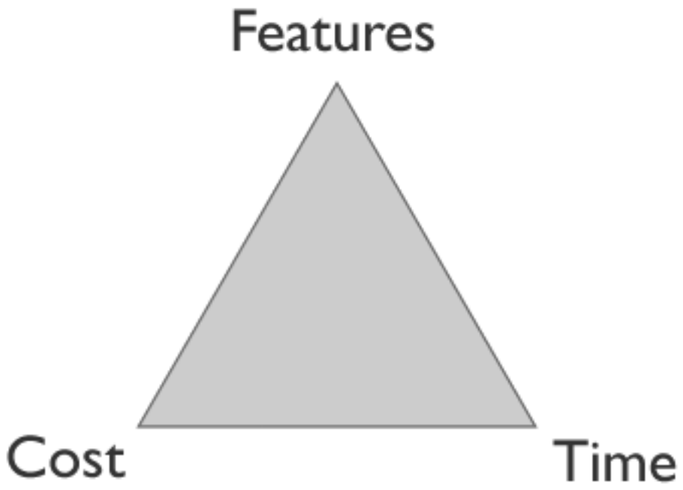
users: purchase and use software product

can be fickle and can misunderstand the product

Course Format

- **Class meetings** ("lectures") to discuss aspects of software engineering and best practices
- **Reading assignments** to reinforce concepts and stimulate critical thinking
- **Small programming activities** to help understand certain key ideas in a more structured fashion
- **Group project** for hands-on experience, and to think about **technical** (project scope) and **social** (team effort) challenges

The Project Triangle



Three factors constrain every project.
You control two and **only two**.

What is a Software Project?

Projects are a balance of three dimensions ... with the goal of producing a successful deliverable.

The Project

- You make proposals (in teams of 3)
  - next week!
- And then vote on which products to "fund"
  - no real money at this point
  - "funding" → a go ahead (suitable in scope, feasibility, interest, etc.)
  - instructors also weigh in

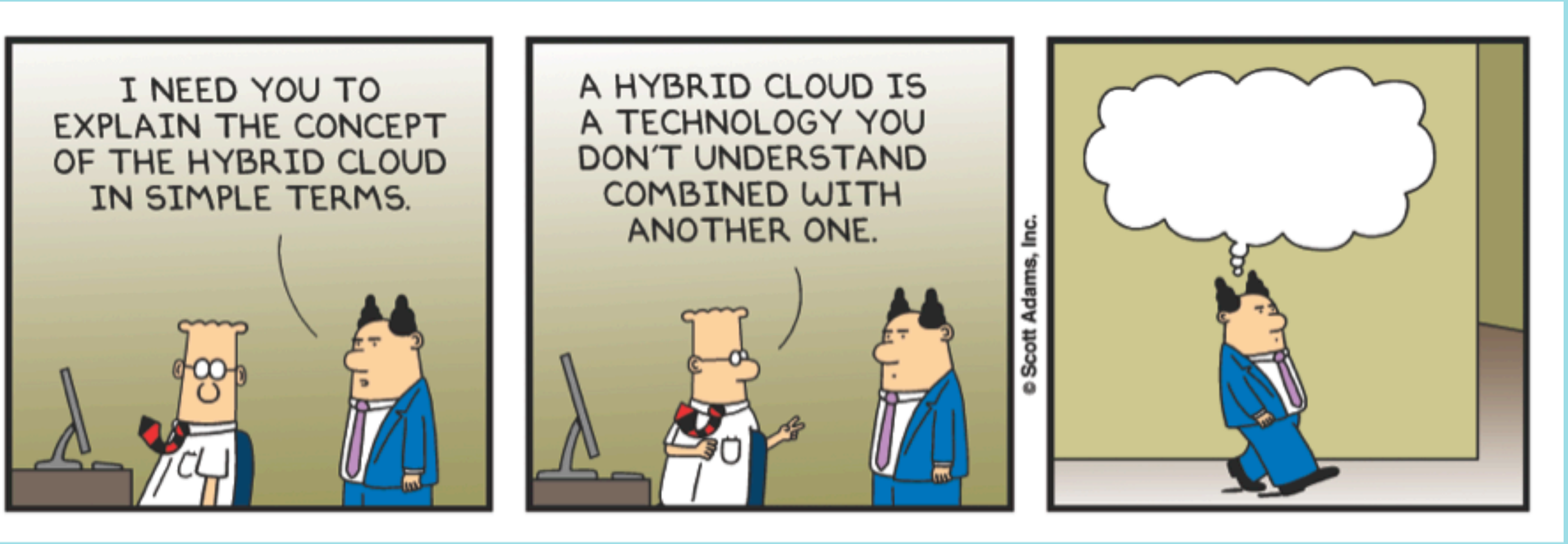
The Project

- Only some of the project proposals will proceed
  - And will be staffed by larger teams (6-7) students
- You then develop the deliverable
  - In stages,
  - And reflecting modern methodologies for effective development
- A subset of the course staff will act as clients/customers
  - A project is successful only if it satisfies the customer!

Important Aspects of the Project

- Proposal
- Requirements
- Design
- Implementation
- Testing, validation, verification
- Documentation
- Customer exposure
- Final deliverable

You choose your tools and frameworks!



The Project

Your opportunity to turn a great idea into a product!

- Prepare a 3-slide, 3-minute product pitch in teams of 3
  - Vision and novelty
  - Architecture
  - Challenges and risks

- Form proposal teams by tomorrow 10:00 a.m.
  - **If you are not in a team by tomorrow morning, course staff will put you in teams (arbitarily!)**
- Proposals due Monday morning
- Pitch your ideas (Tuesday and Thursday next week)
- Vote by Friday next week at 12 noon (rank your choices, team *preferences*)

Project Culture

This is a real project

- We expect you to build a real system
- To be used by real people

Project Culture

This is truly engineering

- Take initiative
- Find and solve problems yourself (as a team)
- Programming is only part of the job
  - But you should be able to do it!
- Good planning, design, team work are all needed for success

Grading

- Project: 50% of course grade
- Programming assignments: 20%
- Reading assignments and in-class activities: 10%
- Exam: 20%

Academic Integrity

- It is simple: *do not cheat*
- If work is to be done individually, do it by yourself
- Do group work in your groups only

Lessons From Projects
(especially ECE capstone projects)

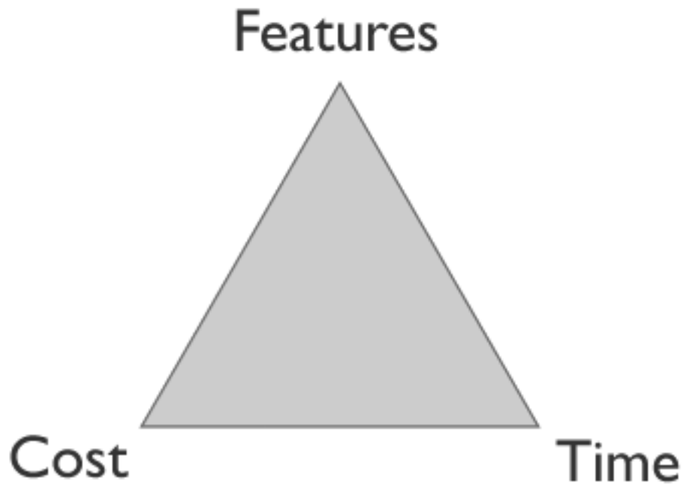
Communication is Key

- Foundation of the success of our team was communication
- Team communication and cooperation are all-important
- Working together (physically) was good
- Well-run and consistently scheduled meetings help a project a lot

Time Management and Scheduling

- We often underestimated tasks. If we had spent more time analyzing each task and breaking it down into more manageable chunks our estimated completion times would have been more accurate.
- Get things done early; don't cram at the end
- Remember you can cut features (triple constraint)
- Don't underestimate the difficulty of learning new programming languages, frameworks, and tools

The Project Triangle



Three factors constrain every project.
You control two and **only two**.

Testing and Coordination

- Thoroughly testing your code and ensuring that your code passes all current tests before submitting is very helpful.
  - We needed a better upfront testing design.
  - We learned (through some pain) to ensure to do small, frequent updates and commits. Failing to do this results in merges that can be a nightmare.

This Sounds Like A Lot of Work! Why Take *This* Course?

What You Can Get From This Course

- See how software is produced, from idea to ship to maintenance
- Get exposure to software development practices in use today
- Get experience collaborating in a team toward a common goal
- Be able to articulate and understand technical ideas

Unique Aspects of CPEN 321

- Cross-disciplinary nature of the subject
- Larger teams
- Propose and work on your own ideas
- Course staff in the "coach" role
- Mistakes along the way are encouraged, not penalized
- Few clearly right/wrong answers
- Plans always change
- Content: software design, testing, etc.

Is this like a co-op position / internship?

It's not in that internships are

- Focused on one role in the team (often dev. or test)
- Requirements, arch, high-level design may be set
- Less opportunity for reflection
- Less generalization (such as from reading papers)
- Mentor may be more focused on results than process and developing you as an engineer

Co-op terms / internships are complementary to CPEN 321

Topics That We Will Discuss

- Software Lifecycle
- Requirements, Use Cases, User Stories
- Software Architecture
  - UML diagrams
  - Design Patterns
- Software Testing
  - From Unit Tests to System Tests

Other Possible Topics

- More on Validation and Verification
  - Static Analysis
  - Symbolic Execution
- Software Configuration Management
- Introduction to Software Project Management

Class Meeting Times

- As the course progresses, we will drop one of the lecture slots in favour of shorter meetings with each project group.
  - Project meeting times will be decided later (after projects and teams are decided)
- Use the allocated lab slots to work on aspects of this course. The lab slots will not be proctored (no TAs!) unless a programming assignment is due - in which case there may be demos in that slot.

piazza.com/ubc.ca/winterterm12016/cpen321/home