## Software Engineering — Curriculum Event

| | | | |
|---|---|---|---|
| Date: | April 4, 2017 | | |
| Department Contact: | Patrick Lam, Director | | |
| Facilitator(s): | Veronica Brown | veronica.brown@uwaterloo.ca | ext. 30196 |
| | Scott Anderson | scott.anderson@uwaterloo.ca | ext. 32509 |
| | Tiuley Alguindigue | tiuley.alguindigue@uwaterloo.ca | ext. 35902 |
| No. of Attendees: | 17 | | |
| Location: | Laurel Creek Room, Delta Hotel, Waterloo | | |

Summary prepared by Veronica Brown

The Software Engineering undergraduate program is offered jointly by the Faculty of Engineering and the Faculty of Mathematics. This event provided an opportunity for faculty to discuss a variety of topics relevant to the program. To enrich the discussions, two students participated as well.

## Agenda

*Times are approximate*

9:00am Breakfast

9:30am State of the SE Program (Patrick Lam)

10:10am What should an SE grad be able to do? (Scott Anderson)
- Ideal Graduate Attributes Brainstorm
- Large group discussion exploring key knowledge, skills, and values we want our students to have by graduation

10:30am New technology trends in the world (ATEs?) (Veronica Brown)
- Based on Ideal Grad, what *technical* attributes should be CORE vs elective
- Review current list – what's missing?
- What criteria do you use to vet potential ATEs?

10:45am Break

11:00am Years 2-3 (Scott)
- Large group discussion – changes needed in year 2? How impact years 2 and 3? Workload issues?

11:30am Student engagement (Veronica)
- Gallery walk to explore ideas related to student engagement

12:30pm Lunch

1:20pm Year 1 changes (Scott)
- Discussion related to current physics requirements

1:45pm CS 247 / SE 465-464-463 sequence (Veronica)
- KSV of Design Wall – each participant is given stickies in three colours (representing KSV), list all the attributes, start grouping on wall then order
- Compare with CS 247 / SE 465-464-463

2:45pm Break

3:00pm Capstone Design Project (Scott & Veronica)

3:30pm ~~Communications (or overflow from earlier) (Scott)~~
- This activity was removed to allow time for other discussions

4:00pm Student Wellness (Veronica)

- Small group discussion with share back
- Group 1: Wellness support on-campus
- Group 2: What is our role as faculty?

4:30pm Wrap-up

4:45pm Adjournment

Prior to the retreat, Patrick Lam provided a schedule with detailed notes related to each topic. We provide the background information at the start of the notes for each section.

# What should an SE grad be able to do?

## Graduate capabilities

**For discussion:** I'd like to come to some sort of consensus on what we expect a Bachelor of Software Engineering to be able to do.

This includes both technical and non-technical skills. I'd also like to have some discussion about the level at which we expect them to perform.

For technical skills, I'd expect to talk about core Computer Science knowledge; specific Software Engineering knowledge; Computer Engineering (notably circuits and feedback control); cognate disciplines like math and stats; and natural sciences (although we'll talk about physics specifically later).

I also want to talk about a liberal education and non-technical knowledge. This includes complementary studies (impact of society on technology; economics; humanities and social sciences) as well as more nebulous aspects like teamwork, leadership, and communication. Ethics should be in there also.

Derek Wright (Graduate Attributes Lecturer) writes:

> My input would be:
>
> - Don't worry about Graduate Attributes - I can do that mapping afterwards.
> - Don't worry too much about how to measure things – we have lots of measurement instruments that are quite good
> - I would like to review whatever you come up with and help tweak the language to improve specificity in the expectations
> - Yes, I will propose a new streamlined process hopefully before May

P.S. How do we measure something like "knowing how to think critically"?

P.P.S. Should we do something different in marketing BSE vs BCS? It is clear that "you can get a PEng" is not, upon exit, where graduates find value.

At the retreat, we had a large group discussion to brainstorm ideas related to an ideal graduate of the program.

- Well balanced – do other things
- Senior roles (not just "worker bees" but understand need for worker bees)
- Out of comfort zone
    - Adventurous
- Not afraid to fail/take risks
- Statistics
- Technically competent
- Not hyper-competitive/ cooperative and collaborative
- Treat as adults/hand-holding
- Comfortable reaching out for help
- Giving attribution/fine lines

## New technology trends in the world (possible ATEs?)

## What's going on in the world

**For discussion:** Are there any ATEs that ought to exist? Are there any topics that we should move from elective to core?

I realize that new-ATE development is most driven by new faculty hiring. But I think it's still a useful discussion to have.

Charlie Clarke writes, informed by his Facebook sabbatical experiences:

1. We should cover the basics of SQL really early. Maybe as part of a 0.25 credit SE 102 course
2. Some exposure to machine learning should be required
3. Targeting a deep understanding of C++ is a real plus of our program, but it needs to be more STL focused [how is CS 247 doing after the 2016 changes?]

---

### How do you add an ATE?

- New faculty's "topic"
- Special topics ← rarely 499 happen SE/ECE
- Need two or three faculty who can teach
- "middle digit" areas → comm, HW, SW
- Pre-enrollment numbers must meet be greater than X to run
- Advanced Topics in "CS/SW" (CS 489)
- Teaching "budget" and can do on own
- Used to have ATE "other"
    - Some now electives, e.g., SYDE
- Teaching area in hiring
- In the end, best research
- Massive list of boutique courses → how impact core [risk]
- Challenge for students is choosing / limiting among choices→ can only take X
- Spring term
- All 4th year (easier to write courses vs. exclusion)

## Themes / synergies

- SE 350 / CS 343 (OS / Concurrency)

## New Topics

- Not “new” but increasing focus: privacy, security
- STAT 206 vs. 230/231
- Distributed dev. (TAKE ECE 454!)
    - Not distributed systems
- Formal verification / software M/C
- Software safety
- Autonomous vehicles ATE
- Privacy
    - The role of dotcom companies in constructing a surveillance state
    - These guidelines could be incorporated into SE490:
    - https://www.priv.gc.ca/en/privacy-topics/privacy-laws-in-canada/the-personal-information-protection-and-electronic-documents-act-pipeda/p_principle/
    - https://blog.torproject.org/blog/technology-hostile-states-ten-principles-user-protection
- Model Checking
- Security
    - currently have elective courses ECE458 + CS458  incorporate some into SE490? incorporate into other courses?
- Data Science / Statistics
    - adding a second stats course
    - is CS adding some electives here?
- Functional Programming
    - is becoming more integrated into mainstream imperative languages
    - more ECE ATEs?

## Years 2-3

Here is a list of core courses that students take in years 2-3.

- 2A: CHE 102, CS 241 (baby compilers); ECE 222 (digital computers); SE 212 (logic); STAT 206
- 2B: CS 240 (data structures); CS 247 (SE principles); MSCI 261 (eng'g econ); MATH 213 (diff eqs+); MATH 239 (combinatorics)
- 3A: CS 341 (algos); CS 349 (UI); SE 350 (OS); SE 465 (QA)
- 3B: CS 343 (concurrency); CS 348 (DB); SE 380 (feedback); SE 390 (FYDP I); SE 464 (design)

Some thoughts (technical):

- What about moving DB to 2nd year/dropping DB? (DB used to help meet CS breadth reqs, which no longer exist; it has successors CS 448, DB impl, and CS 451, Big Data.) See the comment above by Charlie as well as the following comment by Jimmy Lin, CS 451 instructor:

  > I agree with Charlie that the students need to know SQL and the basics of relational algebra early. Whether that is 348 or something earlier, that can be discussed.
  >
  > But in 451, I launch directly into talking about data warehouses - I just assume they know things like primary/foreign key relationships and the difference between an inner and outer join.
  >
  > To guarantee they get this, it's currently CS 348, so from that perspective, it should be a pre-req.
  >
  > SE might be a different bunch from general CS, but if we remove CS 348 as a pre-req for 451, I could conceivably have students walk in without ever having written an SQL statement...
  >
  > -Jimmy

- More stats?
- Concurrency: ECE has added a concurrency course before OS

Some thoughts (non-technical):

- Can we reduce econ/impact? Also see course ratings below.

  ECE 192 LEC, TUT 0.25 Engineering Economics and Impact on Society The course teaches engineering economics and the impact of engineering on the society at large. Important concepts of engineering economics including cash flow diagrams, present worth, quantification of impact costs and rate of return analysis are presented. The course will discuss real-life engineering cases that cover the above aspects and provide a broad perspective on the issues.

Energy supply scenarios and the environment, global energy use and supply, and environmental impacts of engineering projects are discussed. [Offered W, S, First Offered Winter 2019]

Here is the requirement for impact (per Simar):

According to Mark Aagaard, we need 6 hours of impact; we are at about 9 hours. We also plan to talk about impact in ECE 190, which is the concepts course. Impact is also covered now in Engineering design days; and Prototyping Lab in 2B in a practical way. Further, impact is something the students have to consider in FYDP. So there is impact in different forms throughout the program.

**For discussion:** Which changes should we make in years 2-3?

## Potential Changes

- Move CS 348 earlier?
    - Practical?
    - Normalization / de-normalization
    - Check with CS
    - Relational
    - No SQL

  Participants were asked to share what change they would like made. The results are shown below. Note, participants could pick multiple options.

| | |
|---|---|
| Scrap | 0 |
| Leave as is | 2 |
| Earlier | 9 |
| Compress | 12 |
| Overhaul | 13 |

- MSCI 261
    - Modify?
    - Context?
    - Micro-econ of SE
    - Project planning?

## Student Engagement

We certainly have many very engaged students. Data point: at March Break Open House we had 35-40 volunteers (most of any program).

The minimal level of engagement that we ask from our students is that they hand in assignments/projects, write exams and, in first year, do labs. Some students can do fine and even get top marks without even going to lecture. In some sense, Waterloo is not providing them much except for a credential and perhaps some support around the co-op process.

I'd say students are very engaged if they:

- show up at lecture and ask questions;
- go above and beyond for open-ended projects;
- take on student leadership roles or are otherwise engaged with the community;
- work individually with professors on research.

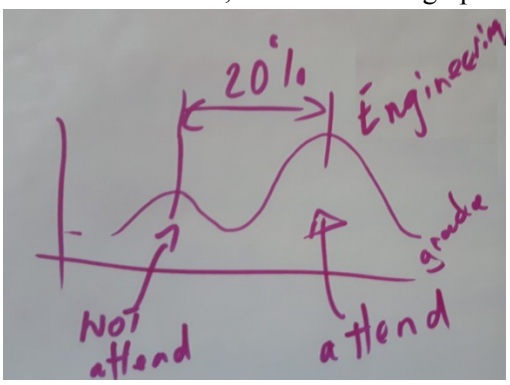
Our students are currently being asked to do the National Survey of Student Engagement, although Patrick White reports that many immediately deleted the email upon reception. (I'd like to think that if I sent the invitation it might be a bit better; in general I think students are more connected with program-level initiatives).

**For discussion:** Should we take measures to improve student engagement? How far can/should we go?

To explore ideas related to student engagement, participants shared their ideas through a gallery walk. Each group discussed a statement about engagement, which was intentionally contentious. They wrote down their ideas then the next group moved to the question to share their thoughts and critique the previous group's comments. Agreement was indicated with a checkmark, disagreement, an X. We rotated four times so groups returned to their initial question for a final opportunity to comment.

### Statement 1: Students are *not* engaged if they don't come to class

- Mostly TRUE ✓ TRUE: 75-80% attendance; FALSE *see graph*




- How to get them to come more?

    - Tell them to
    - Be great ✓(define great)
    - Participation marks? X X
        - For assigned pre-prep activity ✓
        - Quizzes
- Flipped classroom
    - One way to improve participation
- Online offerings
    - We should ✓
    - We shouldn't
- Discussion boards
    - Participation for activity
    - LEARN
    - PIAZZA
- Non-curricular group projects
    - ✓ Sure but doesn't substitute for coming to class
        - Yes it does
- Engaged with
    - materials?
    - Peers?
- Many ways to engage with program – class is one way
- Put it on the prof to make class engaging X ✓
- Make own decisions as adults with respect to class/not coerce
- Come for social aspect
- Depends on teaching and learning style

## Statement 2: It's the student's job to engage with the program not ours!

- It goes both ways ✓✓✓
- Bring in speakers to class ✓ ✓
- Two types on engagement
    - In class ✓✓✓
    - In the department (BBQ, etc.) ✓✓
    - Other events?
- Three types of students ✓✓✓
    - Engaged
    - Attentive but not engaged (may be shy or afraid to be wrong)
    - Not attentive (why not?)
- Prof needs to have an engaging class ✓
  but it is the student's responsibility to participate (not forced) ✓
- Class rep who brings student perspective ✓✓
- Yes, but we should create/encourage a culture of attendance/engagement
- Class reps to encourage students to attend ✓✓

- Incentivize better teaching
    - Teaching awards ✓
    - Don't know how
    - Increased merit weight ✓
- Learning how to make classes more engaging ✓✓
- Research moves for teaching performance?! ✓✓✓
- Students come motivated to be engaged
    - It's ours to lose
- Responsibility
    - Students – learning
    - Faculty – curriculum design, outcomes, interest in learning
- More than just class
- Ultimately the student is responsible for deciding their level of engagement
- Especially want to emphasize engaging with peers

## Statement 3: Our students shouldn't participate in extracurricular activities

- Shouldn't feel obliged ✓
- Internal vs external motivation
- Shouldn't be source of stress ✓✓
- Strongly encourage (especially first years) to participate in some club activities/fun ✓✓✓✓✓✓
- Allows subconscious to absorb learned material
- Un-Programmed activities
- Exercise/sports ✓✓
    - Intramural leagues in ECE/CE/SE
- Unlearn "academics is everything" mentality
    - Learn soft (non-SE tech) skills
- Eat healthy
    - Cooking
- Fear of missing out
- Cali or bust
- Mark-based assessment
    - If struggling, don't participate other than Friday night/Saturday social activities
- Projects such as solar car, but SE-central?

## Year 1 changes

Recently, our students have been doing extraordinarily well in their first-year courses.

| | 1A mean | 1A 1st dec | 1A fails | 1B mean | 1B 1st dec | 1B fails |
|---|---|---|---|---|---|---|
| 2021 | 85.4 | 95.8 | 2 | | | |
| 2020 | 81.3 | 93.0 | 4 | 78.7 | 91.6 | 4 |
| 2016 | 75.5 | 90.3 | 9 | 71.4 | 88 | 22 |

Current 1A courses = CS 137 (Programming Principles); MATH 115 (Linear Algebra); MATH 117 (Calculus 1); ECE 105 (Classical Mechanics); ECE 140 (Linear Circuits); SE 101 (concepts).

Current 1B courses = CS 138 (Data Abst & Impl); MATH 135 (Algebra); MATH 119 (Calculus 2); ECE 106 (E&M); ECE 124 (Digital Circuits)

Starting in 2017: ECE 140 moves to 1B, MATH 135 moves to 1A

Simar Saini writes:

> ECE 106 is a bit harder than PHYS 122 as we cover some topics like boundary conditions not covered in 1st year physics courses. However, these extra topics also make the problems really interesting and relevant. Without understanding boundary conditions; you do get problems in 1st year physics which are erroneous or overly simplified. I have also heard depending on who is teaching it; PHYS 122 may sometimes be not calculus based (Firas, can you please confirm or refute this) while ECE 106 has strong calculus in it and requires lots of deep thinking.

Firas Mansour writes:

> The first 5 weeks of Phys 122 are dedicated to oscillations and waves, the course does E and M. The course is calculus based, however, we cannot cover all the topics we cover in ECE 106. It does cover DC circuits which ECE 106 does not (and does not need to as they see in 1 A). Simar is correct in that it does not cover boundary conditions, an important topic, and adds much depth, however, ECE 106 is unique in that it does so at first year level. Phys 122 does not cover Faraday's Law or induction.
>
> An important differentiator as, Simar mentioned, is the level of the problems which is higher in ECE 106 than in Phys 122.
> Hope this helps,
> PS, There is also Phys 112, the algebra based version of Phys 121.

**For discussion**: ECE 106 is hard and causes student stress. It is a terminal physics course and has no successors except for possible technical electives. Courses with ECE 106 with requisites: ECE 209 (materials), taken by 6 students in 2010-2011; ECE 240 (Electronic Circuits 1), never taken by an SE student; ECE 361 (Power Systems); ECE 375 (EM Fields and Waves); ECE 403 (Thermal Physics)

I asked students about keeping ECE 106 vs replacing with an easier course and perhaps adding a first-year project course.

| | keep 106 | 122 + proj | 122, no proj |
|---|---|---|---|
| 2021s | 25% | 70% | 5% |
| 2019s | 50% | 33% | 16% |

---

- ½ course project?
- Physics easier? ECE106
- Need to buckle down – new to everyone
    - Challenge early
        - Interest
        - Study habits
- Transfers? To ECE
- PHYS 122 – room for DB?
- Stress?? – let students choose
- ECON in 1B?
- Incremental benefit
- ECE benefit with other department
- LAB?
- Another ½ in 1B? – consider co-op
- LAB helped
- Attrition data?
- After discussion most wanted to keep ECE 106; MATH 135 moving to 1A may help workload

## CS 247 / SE 465-464-463 sequence

The three-course sequence SE 465/464/463 (testing/design/requirements) distills the core of the academic subfield of software engineering. SE faculty believe that this is what SE students should be coming here to do. Yet, these courses are unpopular among students, as reflected both anecdotally and in course evaluation numbers.

Engineering Q17: "What is your overall appraisal of this course?"

| | median |
|---|---|
| SE350 | 71 |
| SE380 | 57 |
| SE464 | 57 |
| SE465 | 62 |
| MATH213 | 75 |
| MSCI261 | 61 |
| ECE390 (not SE) | 53 |
| ECE356 (not SE) | 72 |

Math Q7: "Did you find the course interesting?

| | very int | v int + int | not int |
|---|---|---|---|
| SE463 | 7 | 41 | 58 |

```
Did you find the course interesting (very interesting/interesting/not
interesting)

  SE 463 S16=3/14/73, F14=2/2/96, F13=0/38/62, F12=9/43/49,
F11=4/38/58, F10=10/60/29, F09=21/43/36

  CS 348 F16=3/71/26, F15=4/37/59, F14=12/84/4, F13=11/60/29

  CS 247 S16=6/43/52, S15=14/62/23, S14=17/60/24, S13=19/60/22,
S12=7/88/5


  CS 138 W16=43/50/6, W15=46/47/6, W14=60/35/5, W13=35/55/11

  CS 343 F16=61/36/2, F15=64/30/6, F14=60/37/3, F13=59/40/2

  CS 341 W16=36/57/6, W15=50/45/6, W14=20/65/15, W13=45/48/6
```

Course outlines:

- CS 247: Software Engineering Principles
- SE 463: Software Requirements Specification and Analysis [topics]
- SE 464: Software Design and Architectures
- SE 465: Software Testing, Quality Assurance and Maintenance

## SE 463 (Requirements)

First of all, if we are supposed to be training software engineers, we absolutely have to include some content on requirements. Even if students think that it's 2017 and requirements are passé. They're not.

Having said that, this course comes up extremely often as a pain point; it is the most-often cited "don't like" in the 2017 exit surveys. uwflow ratings are abysmal.

The most recent offering (by Jo) asked the students to use their FYDP as the system for which they are eliciting requirements. Derek Rayside has heard positive comments about that choice during the most recent offering as he was teaching the FYDP course. uwflow got the negative comments, Derek got the positive comments.

From uwflow:

> I was lucky enough to take this course with Jo Atlee, who is one of the better SE professors; however, even not Mother Teresa could make a miracle out of this one.

From 2017 exit survey:

> I wish that I knew the things we learned in SE 463 at the beginning of our 4A term when we should have been eliciting the requirements of our FYDP from our clients. Learning them later that term was great but we'd already invested a ton of time into a solution which turns out was incorrect. Perhaps if we'd known the elicitation techniques before hand then we wouldn't have had to restart our FYDP.

I submit that it doesn't need Mother Teresa. Kenny Wong at Alberta definitely is not Mother Teresa and offers a 4-week Coursera course with Coursera ratings of 4.7. (Coursera average is allegedly 4.5 and the lower bound for Coursera might be around 3.9, according to this source.)

I don't know really where to start, but two possibilities are the list of topics and the project. Clearly a requirements course has got to be in the context of some project. We've had projects imposed by the instructor and FYDP-as-course-project in the past. It is true that some FYDPs are less suited for SE 463, but they are probably more suited than students think.

I solicited some thoughts from an SE 2009 grad. Industrial perspective on requirements

It's also possible to vary the amount of formal methods used in this course.

Refresher for those of us who aren't software engineers. Andy Ko writes about requirements and specifications.

**For discussion:** What do we do?

## CS 247/SE 464 (Design and Architecture) overlap

Students often feel like there is a lot of overlap between CS 247 and SE 464, since both talk about design patterns. Faculty disagree.

Jo Atlee writes:

> I would be really sad if design patterns were removed from CS247. For one thing, the OO design patterns are all related to small collections of related classes, and this is exactly the size of program that CS 247 is all about. And the OO design patterns nicely build on the OO design principles that we cover in the course. Design in CS247 nicely progresses from (1) design of individual classes to encapsulate (mostly) data representation, to (2) design principles for OO programs, to (3) design patterns for encapsulating lots of other small design decisions. [Werner agrees]

Werner Dietl writes:

> Many students complained about design patterns being a repeat from SE 247 (or some such) even though we discussed many additional patterns; that architecture in the abstract was too high-level, even though we discussed concrete examples. I discussed several topics I though would be exciting later in the course (like microservices), but already had lost most students. The feedback from the class reps didn't reflect what I see in the written overall feedback. I see a lot of the criticism about my offering of the course was also given for previous iterations.

I discussed various ideas for improving SE 464 with Werner before the retreat. Some of the ideas are fully within the purview of the individual instructor. Points that are more useful for the retreat:

- SE 464 should really deal with actual codebases. Writing code specific to an assignment is going to be too small. We can consider possible synergies between SE 390 miniprojects and SE 464 subject code.
- Design Patterns: it is possible that students think they understand design patterns but don't actually understand some points (e.g. multiple dispatch). Can we evaluate student understanding? Is there some design patterns exercise that we can do that helps students understand that they don't understand? If they actually do understand, then we should remove the material from SE 464.
- Reading code: might have the ability to read code as an explicit goal of the course.
- Too Many Projects: Currently SE464/ECE452/CS446 have an open-ended program design project. This overlaps with SE capstone. It makes sense to have this project for CS and ECE students, who might not have another opportunity to do this kind of thing. We could consider moving the good ideas from this project into SE490. (credit: DR)
- Model Checking (Werner has incorporated things like ArchJava and type systems work for reflecting architecture in code into the course). Text courtesy Derek:

  > If we remove the Gang of Four design patterns and the open-ended program design project from SE464, there will be space for new ideas.
  >
  > Most design patterns are focused on **fitness for future:** why this design will be robust to certain kinds of anticipated future change. **Fitness for purpose** often takes a secondary role in the design pattern literature. One possibility is that we could expand SE464 in the direction of **fitness for purpose**.
  >
  > Model-checking is one technology that can be used for establishing a design's fitness for purpose. We have successfully taught how to use model-checkers in SE464 before: in the summers of 2010 and 2011 we taught Spin and Alloy in SE464.
  >
  > There could be other ideas in the direction of **fitness for purpose**.

Andy Ko on architecture and program comprehension.

## SE 465

My impression: it's not perfect, but putting it in 3A was good. I like what I've done with the course this term—first part is about choosing what to put in test suites; second part is about engineering test suites; third part is about tools.

---

Before discussing this course sequence, participants were asked to consider aspects of design, specifically the knowledge, skills and other elements they want their students to develop.

### Knowledge→ concepts, theories, methodology

- Programming paradigms
- Patters
- Effective use of PCs
- Constraints
- Architect
- Programming
- Correctness
- Design patters
- Design patter and tactics
- Understanding who stakeholders are
- Separation of concerns
- Software architecture
- Requirements
- ER/data modeling
- Models of computation
- Logic (discreet math and correctness)
- "Laws of SW design" e.g. Amdahl, Conway, Lehman
- Normal vs Radical design
- Design for quality attributes

### Skills

- Abstraction
- Generating specifications
- Solve right problems
- Debug, Issue (bug) tracking
- Verification and validation
- Database design
- Code review
- Reqs elicitation
- Read docs
- Should be testable
- Benchmarking
- Dependency mgmt.
- Testing! Early! Often!
- Release mgmt.
- Communication
- Solve problem by thinking
- Modularity
- Cross-platform development
- Use right tools
- Version control
- Incremental development and release
- Collaboration
- "elevator pitch" explain what you do
- Presentation skills
- Declarative thinking
- Problem breakdown
- To synthesize a solution from all of the relevant constraints
- To frame, or re-frame, the problem and objective
- To create alternatives
- To select from those alternatives
- Considering and evaluating design alternatives
- Prototyping
- Lateral thinking
- Model checking

## Other

- Legal: security issues, privacy issues, intellectual property
- Social: impact of software
- Economic: cost of development
- User-centric
- Appreciate code design
- Confidence in programming
- Maintainable, considering maintainability
- Algorithmic analysis
- Colab remotely
- Consider trade-offs

## CS 246/247 (2B)

- CS students hate course because don't like design
    - (oh, it works so I'm done) (value)
- SE much more interested in evaluating design
- CS/SE/CE → all good programs, pick one, be realistic
- SE students really curious about it "what if …"
- People interested in 465/4/3, not just worker bees
    - Distinguish between good and great program
- In CS 246, now in 2A, makes huge difference with maturity for 247 in 2B
    - What take from course very different because variety of interest, reason for taking 246
- Different plans after undergrad – influence (little interest in Grad school)
- "Building" a program vs a whole project
- What is the perception? Expectations?
- What about P.Eng and CEAB? How get P.Eng. in CS?
- 247 overlap with 464 (design and architecture)
    - SE small/med e.g., design patterns
    - 464 project overlap with FYDP
- In 247, implement design patterns
- Overlap in 464 → by the time 247 and coop, maybe too much
    - Depends on your coop? some are less frequent
- Do students know design patterns as well as they think they do?
- 464: perception, already spending a lot of time on what have seen; not enough time on implementation
    - For some, just remember for MT and final
- More practice ( practice design patterns not seen on coop)
- Opportunity: groups of design patterns → student choice
- Where use design patterns in other courses
- 247 → good → why use design patterns and applications
- Patterns in 138
- Are employers interested in how use/know design patterns and when beneficial to apply design patterns
    - Helpful on coop → understanding others' code
    - Over course of coop, being able to recognize where use design patterns
- Shared language
- Because of overlap, shift focus in 464 → compare/contract patterns → critical analysis of patterns

- Architecture
    - How to extend existing
    - Distinguish between architecture patterns and design patterns
    - Patterns and different languages
- Really good @ design single system, what happens when need to integrate across systems
- Needs to go "beyond" coop
    - Risk, varied coop experience
    - How to assess?

## SE 465/464/463

- 465 – Testing - Really well done, wish it was earlier (testing)
    - 465 QA too (not just testing)
- Opportunity – some testing at start of 247 (multiple places in curriculum)
- Where getting QA as it relates to testing? (should be integrated)
    - Perhaps increase weight of testing / QA in other courses?
-
- Formal verification/software M/C → does this need to be integrated across the curriculum?
- 465 and 464 are siloed
    - How better integrate?
- 463 – requirements
    - Have to have if going to be an SE program
    - Most complained about
    - Being turned around
- Students like: spec tools, required techniques, user stories & profiles useful, see use right away
- Students dislike: integration of FYPD → good in principle, took a lot of group time and a lot of work, scale to group size; talk to stakeholders about "menial" stuff, deliverables built on SE if too much
- 463 – course HAS to have a project, so not just a modeling course
- FYDP → helps make less artificial
- Positive → can focus on project, content, experience activities → must learn by doing (e.g. actually talking with stakeholders) BUT trick is to balance against "busy work"
- Positive → differentiates from the "well-defined" problems
- Will take workload issues to heart

## Capstone Design Project

My perception is that things went very well this year.

Derek has worked very hard to increase the quality of the projects. We have a flexible incentive/marking scheme which includes multiple paths to excellence. I want to talk about both the top groups and the average groups.

We gave two first prizes this year: one in the Entrepreneurial category to Dynalist, which is developing software for collaborative, hierarchical to-do lists, and one in the Research category to Manifold, which is working on synthesis of microfluidics circuits. Manifold had a paper accepted to CCECE. We gave third prizes to a group that developed working Scantron-replacement technology and a group that "probably did something publishable" in the computer graphics space. Honourable mentions went to teams that showed a great depth of understanding about sheet music composition and that implemented significant code for image migration in the Xen hypervisor.

It's also important to focus on the average-quality project. Derek has done excellent work here. All else being equal, the groups that I refereed were probably average groups. Derek has gotten the message across that students absolutely have to talk to users if they are doing a technically-straightforward project. That showed up quite strongly here: both of my groups had been deploying their code for about a month and had some actual arms-length users. This definitely improved the quality of their projects and it's what software engineering really should be about: meeting the needs of end users.

The only concern I should mention is that we've noticed a couple of students who have switched out either to avoid the Capstone Design Project, or because they felt that they weren't pulling their weight on the team. I don't think this is a problem, but this is something that should be noted.

**For discussion:** I propose that we ratify what we're currently doing.

---

After a brief discussion, the current approach was supported.

## Communications

**For discussion:** Do our SE graduates have enough experience with both oral and written communication, and in the appropriate forms?

I'd like to reach a consensus on what our students actually need to be able to do in terms of communication. Writing reports is not the same as writing descriptions of pull requests.

Oral communication: students must do a Technical Presentation Milestone (15 minutes), and, in groups, they must orally present their Capstone Design Project (20 minutes). Many students also take a SPCOM class as their communications elective. SPCOM is about non-technical oral communication.

Written communication: I believe that work reports are the only mandatory written communication from our students. Courses may have written deliverables, but none of them are structurally guaranteed. Is this appropriate for their needs? Too much? Wrong form? What about ENGL 210E?

*We did not address this topic at the retreat.*

## Student Wellness

**For discussion:** What more can we do to promote student wellness, particularly mental health?

Software Engineering is a high-workload program. At the March Break Open House, our current students were reassuring the prospective students and their parents that it's doable. Which is true. We have 800+ grads, after all. But SE is not easy.

As I see it, there are three main factors.

- Workload, as we've discussed through the day. As faculty, we have most control over this point. It is the most proximate, but perhaps the least of, students' actual worries. It didn't really come up in exit survey results. The average reported hours per week was 60, although no one actually reported that number; 69% reported more than 60 and 11% reported less than 60.
- Self-imposed expectations and "keeping up with the Joneses". Many students have internalized Cali-or-bust and side projects. They think that they have to do a zillion things to be good students. It's going to come with the territory in terms of the student body that we're getting these days. How can we moderate student expectations and promote more self-compassion?
- Knowing how to get help. What can we do to make sure that everyone knows how to get help when they need it?

An important aspect of student wellness is having friends. I think our students are fairly supportive, and the cohort system undoubtedly helps with this. I don't care if students have friends within the program or not, but I don't want students to have no friends. How do we reach students with no friends?

Derek Wright writes:

> For student wellness, I would like you to consider proactive measures, like teaching sessions on Cognitive Behavioural Therapy and Mindfullness techniques. It’s super easy and there is tons of research showing the benefits.

[PL] SE 101 would be a great time to do this.

We asked participants to form two groups for a small group discussion about support related to student wellness. The groups discussed one of two questions: TWO QUESTIONS. It was an intense conversation and participants indicated an interest in learning more about resources available to gain further training in this area. We have provided a resource list at the end of this section.

## Group 1: Wellness support on-campus

### Student Perspective

- Counselling Services
- Engineering Counselling
- Peer Counselling
- Informal personal support group
- Since the program is small, easy ??? to the program
- Student Success Office
- Student-facing reception/academic advisors → Faculty are accommodating
- Don in dorms
- Faculty counselling services

### Preventive

- Study skills in Student Success Office
- Peer groups
- Mental Health and Wellness
- Puppies and Poets
- Lunch with upper years
- Structured mentoring could be good
- Long wait times (3 weeks)
- What causes the stress
    - Am I doing well academically?
    - Will I get a good co-op?
- The pressure that students put on themselves
- Managing expectations
- "What I wish I knew in 1B"

## Group 2: What is our role as faculty?

- Recognize when a student needs help and what kind
    - Be supportive. Do not become the counsellor
- Make oneself and then students aware of what is available
- What do we do with those that do not make us aware?
    - i.e., they do not come to us
- As a peer student – it is even harder to deal with fellow students that go through difficulty
    - Recognition
    - When it worsens
    - What to do
- Need support as advisor(s)
    - Emotional/psychological cost from dealing with student issues
- When high-achieving students "fail"
- International students

## Campus Wellness - Faculty and staff supporting students

Campus Wellness encourages faculty and staff to consider the following steps when responding to students requiring support.

1. **Recognize** the indicators of mental illness
2. **Respond** to the student in a way that is appropriate to the situation at hand and the existing relationship you have with the student
3. **Refer** the student to the appropriate resources so that they can access the services available

https://uwaterloo.ca/campus-wellness/counselling-services/faculty-and-staff-supporting-students

## Training available to faculty and staff

Campus Wellness offers several mental health and suicide prevention programs for students, faculty and staff. They are described in great detail on their web site.

- Core programs such as QPR (1.25 hrs) and More Feet on the Ground Online (1-1.5 hrs) are our most widely available training programs.
- Programs such as safeTALK (3.25 hrs) and More Feet on the Ground In Person (2.5 hrs) may be of particular interest to student leaders and peer mentors, staff

and faculty leaders (e.g., advisors, Associate Deans), as well as other student services employees.

- ASIST (12 hours) and Mental Health First Aid (12 hours) are our most intensive training programs and may be of particular interest to police services, residence staff, counsellors, nurses, and physicians.

https://uwaterloo.ca/campus-wellness/about-campus-wellness/training-and-internships

## Athletics and Recreation – Wellness

A recent change on-campus that is also worth noting is the focus on Wellness at Athletics and Recreation. The department's web site now has online recourses as well as other initiatives to support student wellness. An example is the Weekly Warrior habits site, http://www.varsity.uwaterloo.ca/news/2016/1/4/WR_0104160858.aspx, which provides suggestions for a healthy lifestyle.

# Bike Rack and Resources

## Bike Rack

The Bike Rack represents a list of topics that were raised at the retreat that were not related directly to the activity. They were documented for future consideration.

- Gender diversity in the program (and other forms of diversity)
- Project (! = class) work
- Attitude of risk aversion
- Admissions
- 23 dropout → switch to other unis or totally out of uni
- What would it mean to be an "elite" program? Smaller, individual attention?
- Fine lines of attribution

## Resources

Provided by Patrick Lam before the retreat.

- SE 2017 partial exit survey: results minus text and salaries
- Desiye Collier (class of 2015) on SE culture: [google doc]
- Bilal Akhtar (class of 2019) on imposter syndrome: medium
- Industrial feedback

At the retreat, the facilitators shared a few concepts from the literature that might be of interest to participants.

Grow, G. O. (1991). Teaching learners to be self-directed. *Adult Education Quarterly, 41* (3), 125-149. doi:10.1177/0001848191041003001

Moore, M. G. (1989). Editorial: Three Types of interaction. *American Journal of Distance Education, 3*(2), 1-6. doi:10.1080/08923648909526659