SE490+CS493 Capstone Design Spring 2021

Prof Derek Rayside

drayside@uwaterloo.ca

August 21, 2021

NOTE: please include 'capstone' in email subject, or consider using your team's channel in MSTeams. Meetings: https://calendly.com/derek-rayside/30min https://discord.gg/jWJEnxxn

Overview

SE490 Course Description

> Continuing from SE 390, students undertake a substantial customer-driven group project. Project groups establish and maintain project control processes, delivering a series of iterations on their initial SE 390 prototype. Adaptive methodologies are encouraged and supported.

https://ucalendar.uwaterloo.ca/2021/COURSE/course-SE.html#SE490

CS493 Course Description

> Students work in teams on substantial open-ended computer science problems as part of the CS 493/494 course sequence. Lectures describe project management fundamentals and ethical and legal issues in computing. Students form teams, select projects, define project goals, perform risk assessment, establish a project plan, and develop a prototype. Possible project topics can include development of software systems, analysis of extensions to existing systems across the field, and experimental computer science.

https://ucalendar.uwaterloo.ca/2021/COURSE/course-CS.html#CS493

Learning Objectives The capstone project is intended to embody essentially all of the learning objectives of the undergraduate degree. Learning objectives have been identified by the Canadian Engineering Accreditation Board (CEAB), the Canadian Information Processing Society (CIPS), and the UWaterloo Software Engineering Curriculum Committee. These are listed in the *Handbook* §1.

Handbook §1

URLs & Repositories

Course Website	https://ece.uwaterloo.ca/~se_capstone/
Course Notes (Handbook)	https://ece.uwaterloo.ca/~se_capstone/se-capstone-handbook.pdf
Project Metadata	https://git.uwaterloo.ca/secapstone/se2022-490.git *(you need to clone this)*
Course Discussion	https://discord.gg/jWJEnxxn
Meetings	https://calendly.com/derek-rayside/30min *(when in the meeting, click on Participants and invite Derek or he won't know that you are in the meeting)*

Contribute Cover Art!

Calling all artists! Please contribute your artwork for the cover of our class abstract booklet. We have used student artwork on the cover for the last few years.

See past abstract booklets on the course website.

Project Evaluation

Handbook §14

Project evaluation is aligned with the learning objectives and is discussed in *Handbook* §14. A central premise is that the software should perform its intended function properly and in a unified way. Generally speaking, capstone projects should, in all relevant ways:

- exemplify the learning objectives;
- demonstrate the skills expected of a graduate;
- make appropriate trade-offs and judgments; and
- not suffer serious oversights.

Handbook §14

Marks are generally assigned to teams rather than individuals. The *teamwork* grade may involve peer evaluation that produces some individual variation in final grades.

In extreme cases, the contributions of individual students might be assessed individually.

Marks are organized as follows:

Facet		*Weight*	*Due Date*
Weekly Learning Activities		20%	
· bootup	2%		May 16
· effort-based assessment	18%		9 out of 12 weeks
Weekly Project Progress		20%	10 out of 12 weeks
· effort-based assessment			
Teamwork		10%	
· quiz	4%		May 23
· health + process self-assess	2%		June 6
· health + process self-assess	2%		~~July 17~~ *July 24*
· reflection	2%		August 5
Peer Feedback		10%	
· activity #1	2%		~~June 13~~ *June 23*
· midterm (to two others)	4%		~~June 20–27~~ *June 24–30*
· activity #2	2%		~~July 24~~ *July 31*
· final	2%		Aug 7–16 (exam period)
Midterm Demo		10%	~~June 20–27~~ *June 24–30*
Final Demo		30%	Aug 7–16 (exam period)

You may choose to do your final demo before the final exam period.

Bootup

May 16

1. git clone https://git.uwaterloo.ca/secapstone/se2022-490.git
2. cd se2022-490
3. mkdir teams/team-OurTeamName
4. cp team-template/*.* teams/team-OurTeamName/
5. userids.txt — uw userids
6. meta.tex — project name, *etc.*
7. abstract.tex — undecided? list ideas you are considering
8. activities.md — list of 15 potential activities you might do
9. scrum0.md — identify initial goals
   (see handbook for suggestions; feel free to include alternatives)
10. make a merge request with your files

Weekly Learning Activities

Everyone gets full marks for making an honest effort + follow through.

> *If you follow through an action, plan, or idea or follow through with it, you continue doing or thinking about it until you have done everything possible.*

https://www.collinsdictionary.com/us/dictionary/english/follow-through

What we mean here by *follow through* is that you address any points or issues raised by your TA. The point of the learning activities is to stimulate your thoughts and your project and your interaction with your TA. So the learning activity might be the start of a conversation, not the end of one. If there are interesting ideas or perspectives that arise from the activity, then you should follow through and address them.

If there are challenging cases where your work lacks honesty, effort, or follow-through, then your TA can ask the instructor for help in assessing part marks.

Remember that the learning activities are not the project. They complement the project, and help you get more out of the educational experience. Let's consider the project itself next.

Weekly Project Progress

Everyone gets full marks for making an honest effort.

The purpose of these marks is to help keep you on track. For years students have said, at Symposium Day, that they would appreciate if we could find a way to help them stay on track through the summer. Let's discuss some aspects of honesty and effort:

- The normal model for course workload at UW is 10 hours/week.
  - For example, in a regular technical course: 3 hours lecture + 3 hours lab + 1 hour tutorial + 3 hours studying.
  - Other universities, such as McGill and MIT, explicitly rank different courses based on the number of credit-hours per week. So then meeting graduation requirements is not just a question of having enough courses, but also of having enough credit-hours, since different courses are weighted differently. At Waterloo courses are supposed to be more uniformly sized.
  - The usual arrangement for most teams in most weeks is something like this:
    * 1 hour for a learning activity (e.g., watching an old project video and writing a response)
    * 1 hour for reading course announcements, notes, etc.
    * 1 hour for meeting with TA
    * 1 hour for team sync-up/planning meeting

    * 6 hours for actually working on your project (designing, analyzing, programming, etc)
  - Some weeks some teams will have different arrangements. For example, if you want to patch the Rust compiler you might need to learn the Rust language, and so you might devote significant time to learning activities on that. Those learning activities should have some structure and some identifiable output. For example, perhaps you port some ECE459 assignments to Rust. Perhaps you complete a Rust tutorial.
- SE463 deliverables are not part of SE490, although they are part of your overall project.
  - Many reputable universities would consider it dishonest to submit the same work for credit in two different courses.
  - There are opportunities for other requirements activities that you can do as part of SE490+CS493 that are not SE463 deliverables.
- Natural variation occurs, and if we are honest then we acknowledge it. It's not going to be a uniform effort every week.
- Scrum self-assessment of effort in the past week:
  - above average > 10 hours
  - average ~10 hours
  - below average < 10 hours
- Full marks: make an honest effort over the term.

THERE IS A GRADING PROGRESSION through the capstone courses:

- We start in SE390 grading you purely on effort.
- Now in SE490+CS493 the scale slides towards effort (weekly progress) and technical assessment (final demo).
- For Symposium Day next March the assessment is on technical merits and real world results.

Project Demos: Midterm + Final

The general goal is to have software that users can try by the end of the term, while applying good software engineering practices. Then you can gather user feedback over the fall coop term and revise your project accordingly next winter.

For some projects this goal doesn't make sense, and that's ok. Discuss with your TA or instructor.

There is some description of this assessment in the Handbook. The best way to callibrate yourselves is by watching old project videos. If you have questions about what to do next in your specific project, then ask your TA or instructor.

University Policies

Intellectual Property: UWaterloo has the (fairly unique) policy that intellectual property is owned by its creators (rather than by the university). The university has resources to help you commercialize your project (if desired), as well as local incubators such as Velocity.

https://uwaterloo.ca/secretariat/policies-procedures-guidelines/policies/policy-73-intellectual-property-rights
https://uwaterloo.ca/secretariat-general-counsel/faculty-staff-and-students-entering-relationships-external

Academic Integrity: In order to maintain a culture of academic integrity, members of the University of Waterloo community are expected to promote honesty, trust, fairness, respect and responsibility.

http://uwaterloo.ca/academicintegrity/

AccessAbility: AccessAbility Services collaborates with all academic departments to arrange appropriate accommodations for students without compromising the academic integrity of the curriculum. If you require academic accommodations, please register with AccessAbility Services at the beginning of each academic term.

https://uwaterloo.ca/accessability-services/

Grievance: A student who believes that a decision affecting some aspect of his/her university life has been unfair or unreasonable may have grounds for initiating a grievance. When in doubt please be certain to contact the department's administrative assistant who will provide further assistance.

Policy 70, Student Petitions and Grievances, §4, http://secretariat.uwaterloo.ca/Policies/policy70.htm

Discipline: A student is expected to know what constitutes academic integrity to avoid committing an academic offence, and to take responsibility for his/her actions. A student who is unsure whether an action constitutes an offence, or who needs help in learning how to avoid offences (*e.g.*, plagiarism, cheating) or about rules for group work/collaboration should seek guidance from the course instructor, academic advisor, or the undergraduate Associate Dean.

http://uwaterloo.ca/academicintegrity/

For information on categories of offences and types of penalties, students should refer to Policy 71, Student Discipline, http://secretariat.uwaterloo.ca/Policies/policy71.htm
For typical penalties check Guidelines for the Assessment of Penalties, http://secretariat.uwaterloo.ca/guidelines/penaltyguidelines.htm

Appeals: A decision made or penalty imposed under Policy 70 (Student Petitions and Grievances) (other than a petition) or Policy 71 (Student Discipline) may be appealed if there is a ground. A student who believes he/she has a ground for an appeal should refer to Policy 72 (Student Appeals).

http://secretariat.uwaterloo.ca/Policies/policy70.htm
http://secretariat.uwaterloo.ca/Policies/policy71.htm
http://secretariat.uwaterloo.ca/Policies/policy72.htm

Reconciliation

We acknowledge that the University of Waterloo is on the traditional territory of the Neutral, Anishinaabeg and Haudenosaunee peoples. The University of Waterloo is situated on the Haldimand Tract, the land promised to the Six Nations that includes ten kilometres on each side of the Grand River.

https://uwaterloo.ca/arts/about-arts/territorial-acknowledgement

Figure 1: Contemporary map of the original Haldimand Tract and the remaining Six Nations Territory (red).

Announcements

Course announcements will be posted in Discord first, then copied here for archival purposes.

May 13

Project Selection Advice

Four good criteria for project selection:

1. There is a defined group of users.
2. You have access to those users.
3. Their needs are at least partially documented by others.
4. Their needs are not currently met by existing solutions.

Weekly TA Meetings

May 27

PURPOSE: To keep you on track. For years students have said at graduation that they wished we had a way to better help them stay on track during SE490. This is just the second year we've had TAS, and so this is what we're trying to address this historical student feedback. Your suggestions for improving the process welcomed.

GENERAL GUIDELINES:

Ask Derek for exceptions. Maybe someone is in a very different time zone *etc.*

- Everyone should attend.
- Make the meetings as meaningful and as short as possible.
- Cameras on.
- Submit scrumX.md file at least two hours before meeting.
- Assign your TA as a reviewer for scrumX.md in Git.

Code Review Tips

May 27

OBJECTIVES. There are three main objectives for code reviews. Your review may address one, two, or all three of these objectives.

Writing guidelines for code reviews: https://phauer.com/2018/code-review-guidelines/

- *Code improvement*
  - Comprehensive checklist: https://www.michaelagreiler.com/code-review-checklist-2/
  - Focus on one or two areas of the comprehensive checklist in your review:
  - Implementation
  - Logic Errors / Bugs
  - Error Handling + Logging
  - Usability & Accessibility
  - Ethics & Morality
  - Testing & Testability
  - Performance

  - Dependencies
  - Security & Data Privacy
  - Readability
- *Learning*
  - Learning about the code across the organization is important.
  - Write what you learned.
  - Write questions that you have about the code.
- *Teambuilding*
  - Code review is a chance to interact with each other.
  - Write something nice: what is good about this code?
  - Call the author to discuss.

Commit Size. Researchers have measured that successful organizations often have commit sizes around:

- 10 lines for small commits
- 25 lines for large commits

Responsiveness is a key factor to successful code reviews. Researchers have measured that top teams aim to respond:

- within 1 hour for smaller code reviews
- within 5 hours for larger code reviews

Criteria for Approval.

- Does this change make the code better? Then you can approve it.
- It doesn't need to be perfect.
- If this change makes the code better, but the review discovers other major issues, then make a new ticket/issue for those issues.

Selecting Reviewers. There are a variety or reasons to select a reviewer, including:

- Someone who already knows this part of the code.
- Someone who should become more familiar with this part of the code.
- Someone who is familiar with code that depends on this code.
- Someone who is familiar with code that this depends on.

May 29

Weekly Workflow

1. Work on project
2. Do learning activity
3. Update scrumX.md with progress
4. Create merge request
5. Assign TA to review merge request
   (at least two hours before weekly meeting)
6. Weekly meeting

Midterm Demos

June 15
This information will be duplicated on the course website and on LEARN.

Demo:

- 15 minutes demo
- 15 minutes discussion
- 30 minute time slot = 15 min demo + 15 min discussion
- 3–5 slides + software demo
  - Opportunity/Problem
  - Architecture overview
  - Team process + health summary
  - Current status
  - Objectives/Plan

Learning activities for the 3 weeks June 14–June 30 can be preparing for the midterm demo. If you do other learning activities, in this time, you may claim them in subsequent weeks instead.

Scheduling:

- Thursday June 24 to Wednesday, June 30th.
- Common case: Your usual TA meeting time.
- Exception 1: Your usual time conflicts with another group's usual time. See the spreadsheet (or talk to your TA) for your alternate time.
- Exception 2: You would prefer to bump your usual time on Thursday, June 24 or Friday, June 25th, into the next week. See the spreadsheet (or talk to your TA) for your alternate time.
- Spreadsheet: https://docs.google.com/spreadsheets/d/1ucyh43PrZSXYC5SDkHMfQ23A-eSw8ZNDxX323pue8vM/edit?usp=sharing

Peer Activity #1: The syllabus has peer activity #1 worth 2%. The original deadline was June 13, now extended to June 23.

We recommend that you choose feedback on practice demo as your peer feedback activity — but you can choose something else if you prefer. This is a team activity, and you will write a file in the directory of the team you are giving feedback to. You have been paired with another team (below). You will give each other feedback.

- Axon + LOI
- SpongeFabric + Vulcan
- scena360 + coachella
- InsufficientlyCaffeinated + inliner
- QUAIL + TheWheelerz
- KRAM + Hydra

- moss + Localised
- Houdini + Agnes + rhino *(triplet due to odd number of teams)*
- LacusLabs + r2
- VoxPopuli + ChickenNuggets
- DiDiDaJi + fnord
- DotDotDash + VivaLaZeez
- alarm + uConverse
- AutumnLeafStudio + BSD

Midterm Peer Feedback:

- The syllabus has midterm peer feedback for 4%, due during the midterm demo period.
- You will give feedback to two different teams (2% each).
- You will do this individually in LEARN.
- You can choose which two demos you want to attend.
- You may also ask questions and give feedback verbally during the discussion period after the demo.
- *You MUST select different teams than your teammates.* So if there are 4 of you on your team, then collectively you will see 8 different demos, since each of you will see two different ones. The depth of your learning in these courses comes from your project. The breadth comes from learning from your peer's projects.

Summary of Todos:

- (Team) confirm your demo time with your TA — see spreadsheet
- (Team) schedule practice demos with your assigned practice team
- (Team) plan which teams you will provide individual feedback to during the demos (remember: you MUST select different teams to give feedback to than your teammates do)
- (Team) prepare demo
- (Team) give feedback to practice team on their practice demo (in Git)
- (Team) receive feedback from practice team on their practice demo
- (Team) present your demo
- (Individual) attend two other team's demos
- (Individual) provide written feedback for those two teams in LEARN

Team Health + Process Self-Assessment #2

July 20
This information will be duplicated on the course website and on LEARN.

Deadline extended to July 24. Extensions available on request.

- *Spotify Health Check*
  - Do this individually in LEARN.
  - Compare answers.
  - Identify any interesting outcomes.
  - Brief writeup in Git: health2.txt
- *SCRUM Process Assessment*
  - One person can complete this in LEARN for the team.
  - Identify areas for possible improvement.
    * A subset of the things your team isn't doing.
    * Some of the things your team isn't doing aren't relevant.
    * Some of the things your team isn't doing might be good ideas.
  - Discuss results with team and prioritize potential improvements. Brief writeup in Git: process2.txt

Peer Activity #2

July 20
This information will be duplicated on the course website and on LEARN.

Peer activity #2 is an independent 2% line item on the syllabus.
It is distinct from the learning activities.
Deadline extended to July 31.

Team Pairings:

alarm + BSD
AutumnLeafStudio + Agnes
Axon + Hydra
ChickenNuggets + uConverse
coachella + VivaLaZeez
DiDiDaJi + ProfJayDolmage −−− odd number of teams
DotDotDash + Vulcan
Houdini + Scena360
Localised + KRAM
LOI + fnord
QUAIL + InLiner
rhino + r2
SpongeFabric + moss
TheWheelerz + InsufficientlyCaffienated
VoxPopuli + LacusLabs

Possible activities:

- §8.6 Peer Design Exploration
- §8.7 Peer Design Review
- §11.13 Peer Usability Review
- §8.5 Privacy By Design Review
- . . . your better idea of what you want feedback on . . .

Team Reflection Due August 5th

August 2
This information will be duplicated on the course website and on LEARN.

Briefly (1-3 sentences each) discuss:

- Your team's experience with the first health+process self-assessment. [Google health + Joel test process]
- Your team's experience with the second health+process self-assessment. [Spotify health + SCRUM process]
- Anything else that your team has *actually* found helpful.
- Anything else that your team thinks *might* be helpful.

Deadline is nominally August 5th. No late penalty so long as completed by your final demo.

Preliminary Feedback on Final Demos

August 9

This information will be duplicated on the course website and on LEARN.

Post a Demo Video for Asynchronous Peer Feedback Some students have asked for the opportunity to see your demo video asynchronously in order to give you peer feedback. Post a link in the Discord general discussion channel if you wish (encouraged but not required).

Communication Issues

- Differences from midterm.
- Be clear about existing **libraries** you are using.
- Remember to include (at least briefly) important aspects of your project **context** that you had previously mentioned at your midterm demo.
- Identify what your possible **symposium targets** are.
  - Maybe at this time you have multiple candidate targets, and you'll commit later when you have more information. That's ok.
  - Note that the peer feedback sheet specifically asks the audience if they understand what your goals are.
- Situate your project in the existing technical landscape — how is your work different (or similar) to other work.
- End on a summary slide — not a 'questions?' slide. This reminds the audience what your project is about and stimulates productive discussion.
- *Measure:* use numbers where possible. For example:
  - User activities: how many users? how long was the activity for each user?
  - Latency: measure it.
  - Make some graphs and charts.

Testing / Validation / Verification

- How do you know that you are building the right thing? Something that solves a problem. Something that users want. Have you done user activities (if applicable) to validate your direction?
- How do you know that you have built it right? Testing or other verification techniques, etc.
- SE101 could be a validation opportunity — speak with Victoria.

Specification & Requirements Clear statements like the following might be applicable to your project:

- Our spec contains the following features/aspects/etc that we have not yet implemented ...

- Our spec defines three levels of response for our system. Every input to the system produces a response at one of these levels.

It would also be good to address the following questions:

- How does the current implementation diverge from the spec you created in SE463?
- It is expected that there would be some things in the spec that are not yet implemented.
- Are there things that are implemented differently than how they are described in the spec?
- Are there extra things that have been implemented but are not in the spec?

Dataset Characterization

- What datasets are you working with?
- How big are they? What's in them? Where did they come from? What have they been used for?
- Why are these datasets appropriate for your uses? What other interesting datasets exist in this area?

Blacklist/Whitelist — find better terms

- These terms are often technically unclear. Certainly that is the case in some of your projects.
- These terms are now understood to be socially unacceptable.
- Find better terms that are clearer and more acceptable.

Privacy

- The major ethical concern for engineers is public welfare. One of the main public welfare concerns for software engineers is data breaches. Data breaches can cause significant loss of user's money and time, and in some cases have lead to loss of life, jobs, relationships, *etc.*
- The best way to protect user data is to not store it.
- Is it possible to keep the user's data just on their local machine / phone? Does it really need to be copied to a server?
- Are user accounts really necessary? Can the software be used anonymously?
- Does the raw data need to be stored? Can a summary of the data be stored instead? Can a cryptographic hash be stored instead? *etc.*
- What benefit does the user get from the existence of their account? Is it primarily for developer convenience or for an exploitative business model?

Engagement with SE101 or SE390 this fall

- SE101 and SE390 this fall are potential opportunities for you to conduct user activities or find collaborators.
- SE101 will be taught by Victoria Sakhnini
- SE390 will be taught by Krzysztof Czarnecki

Optimization

- The First Rule of Program Optimization: Don't do it. The Second Rule of Program Optimization (for experts only!): Don't do it yet." — Michael A. Jackson
- Programmers waste enormous amounts of time thinking about, or worrying about, the speed of noncritical parts of their programs, and these attempts at efficiency actually have a strong negative impact when debugging and maintenance are considered. We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil. Yet we should not pass up our opportunities in that critical 3%. — Donald E. Knuth
- Before you try to optimize:
  - Measure the current performance.
  - Determine which parts of the software/system are the bottleneck. It might not be what you first thought.
  - Determine the fastest it could possibly be. For example, if you are trying to optimize latency over a wide area network, then also measure the latency over a local area network.
  - Identify why this performance matters for the user. For example, there are well known human perceptual thresholds for how fast the brain can process sound and vision. What extra things can the user do if the performance is better?

Design Process: Phases of Convergence and Divergence Design usually goes through phases of:

- *convergence*
  - focus on building something
  - narrow the set of possibilities being considered
- *divergence*
  - explore new ideas
  - broaden the set of possibilities being considered

Moving towards a final demo is a convergent phase, where the focus is on developing a concrete idea. Afterwards is a divergent phase, where new ideas are explored.

Refining your success targets

- Many projects are at a point where it's time to stand back a bit and think about refining their targets.
- Some teams need to identify possible paper publication venues for their ideas. Different venues will have different points of emphasis and be looking for slightly different things.
- Some teams are thinking about whether to focus on user acquisition, user studies, or technology development.