Administrivia

- **Proj1_Milestone0** due **9/16 11:59pm EST**
  - team name
  - project proposal
- **Vote for idea** due **9/17 11:59pm EST**
- **About Lab**
  - tutorial by TA
  - lab task, submitting by Friday (participation)
  - Q&A, group meeting (not required to stay for the whole session)
Why did you pick this class?

• I'm planning to work in industry after graduation
• To get strong first-hand experience
• improve my project management and programming skills
• To find a way to better understanding and editing the code written by others
• I am interested in web page design.
• It is a hot topic!
What do you want to learn?
Office Hour

- 68 replied
- Friday 4-5pm EST
- By appointment
Learning Goals

• Recognize the Importance of process
• Understand the difficulty of measuring progress
• Use milestones for planning and progress measurement
• Understand backlogs and user stories
Software Engineering?

„The Establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.”

[Bauer 1975, S. 524]
2013

• 2M people working on 300K software projects in the US
• 1/3 - 2/3 exceed schedule and budget targets before delivery
• Of the most expensive software projects, about half will eventually be canceled for being out of control.
Software projects succeed or fail based on how carefully they are planned and how deliberately they are executed.
Process
How to develop software?

1. Discuss the software that needs to be written
2. Write some code
3. Test the code to identify the defects
4. Debug to find causes of defects
5. Fix the defects
6. If not done, return to step 1
Software Process

The set of activities and associated results that produce a software product
Example of Process Decisions

- Writing down all requirements
Example of Process Decisions

• Writing down all requirements
• Require approval for all changes to requirements
Example of Process Decisions

- Writing down all requirements
- Require approval for all changes to requirements
- Use version control for all changes

**In case of fire**

1. git commit
2. git push
3. leave building
Example of Process Decisions

• Writing down all requirements
• Require approval for all changes to requirements
• Use version control for all changes
• Track all reported bugs
• Review requirements and code
• Break down development into smaller tasks and schedule and monitor them
• Planning and conducting quality assurance
• Have daily status meetings
• Use Docker containers to push code between developers and operation
Example of Process Decisions

• Writing down all requirements
• Require approval for all changes to requirements
• Use version control for all changes
• Track all reported bugs
• Review requirements and code
• Break down development into smaller tasks and schedule and monitor them
• Planning and conducting quality assurance
• Have daily status meetings
• Use Docker containers to push code between developers and operation

How to develop software?

1. Discuss the software that needs to be written
2. Write some code
3. Test the code to identify the defects
4. Debug to find causes of defects
5. Fix the defects
6. If not done, return to step 1
Trashing / Rework

Productive Coding

Process: Cost and Time estimates, Writing Requirements, Design, Change Management, Quality Assurance Plan, Development and Integration Plan

Percent of Effort

Project beginning | Time | Project end

100% | 0%
Productive Coding

Trashing / Rework

Percent of Effort

0%

100%

Project beginning

Time

Project end
Survival Mode

• Missed deadlines -> "solo development mode" to meet own deadlines
• Ignore integration work
• Stop interacting with testers, technical writers, managers, ...
Productive Coding

Trashing / Rework

Percent of Effort

Project beginning

Time

Project end
26. I saw at least one software project fail and (believe to) know the main reason

More Details

- yes: 31
- No: 50
Distance Learning: Day 4 😊

Jeff Dean (@JeffDean) @JeffDean · 19h
Replying to @MrsNewell22 and @ChiefScientist
At least he's hanging in there..

Paul Cantrell @inthehands · 18h
Remote learning has turned so many people's lives

Paul Cantrell @inthehands · 18h
Some have gone batty

Alexy Khrabrov and 2020 others @ChiefScientist
some=>all;)
Example process issues

• Change Control: Mid-project informal agreement to changes suggested by customer or manager. Project scope expands 25-50%

• Quality Assurance: Late detection of requirements and design issues. Test-debug-reimplement cycle limits development of new features. Release with known defects.

• Defect Tracking: Bug reports collected informally, forgotten

• System Integration: Integration of independently developed components at the very end of the project. Interfaces out of sync.

• Source Code Control: Accidentally overwritten changes, lost work.

• Scheduling: When project is behind, developers are asked weekly for new estimates.
Percent of Effort

100%

Trashing / Rework

Productive Coding

Process

Project beginning

Time

Project end

The Edward S. Rogers Sr. Department of Electrical & Computer Engineering
UNIVERSITY OF TORONTO
Real world cases

Organizations that have explicitly focused on improving their development processes have, over several years, cut their time-to-market by about one-half and reduced their costs and defects by factors of 3 to 10.

5 yr, cost -75%, time - 40%, defects - 90%

8 yr, cost -50%, defects - 75%
Planning
Task: Estimate Time

• a web application of Trip guide (booking, scheduling, route planning...)

Estimate in 8h days (20 work days in a month, 220 per year)
Revise Time Estimate

• Remember the GIS system experience?
• Is GIS similar/different/easier/more challenging/reusable?
• How much design did you do?
• Break down the task into ~5 smaller tasks and estimate them.
• Revise your overall estimate if necessary
• 2 Types of Projects
  • Projects having an accurate target, technical inquiry and deadlines.
  • Projects having a general idea and no accurate visualization of further development, like products for startups or Time & Material projects.

https://www.codica.com/blog/how-to-get-better-estimates/
It is important to concentrate on the scale of complexity, not the amount of further work.
Measuring Progress?

“I’m almost done with the app. The frontend is almost fully implemented. The backend is fully finished except for the one stupid bug that keeps crashing the server. I only need to find the one stupid bug, but that can probably be done in an afternoon. We should be ready to release next week.”
Measuring Progress?

• Developer judgment: x% done
• Lines of code?
• Functionality?
• Quality?
Milestones and Deliverables

• Making progress observable, especially for software
• Milestone: clear end point of a (sub)tasks
  • For project manager
  • Reports, prototypes, completed subprojects
  • "80% done" not a suitable milestone
• Deliverable: Result for customer
  • Similar to milestone, but for customers
  • Reports, prototypes, completed subsystems
Project Planning

1. Identify constraints
2. Estimate project parameters
3. Define milestones
4. Create schedule
5. Check progress
   - Done?
     - yes: Technical review
     - no: Reestimate project parameter
6. Reestimate project parameter
   - new feature requests
7. Refine schedule
   - Problem?
     - yes: renegotiate constraints
     - no: Check progress
9. renegotiate constraints
10. activities begin
11. every 2-3 weeks
12. Abort?
Gantt Diagrams
Brief intro to Scrum
Elements of Scrum

• Products:
  • Product Backlog
  • Sprint Backlog

• Process:
  • Sprint Planning Meeting
  • Daily Scrum Meeting
  • Sprint Retrospective
  • Sprint Review Meeting
Product Backlog/Sprint Backlog

• The product backlog is all the features for the product

• The sprint backlog is all the features that will be worked on for that sprint. These should be broken down into discrete tasks:
  • Fine-grained
  • Estimated
  • Assigned to individual team members
  • Acceptance criteria should be defined

• User Stories are often used
Backlog – information radiators
Scrum meetings

• Sprint Planning Meeting
  • Entire Team decides together what to tackle for that sprint

• Daily Scrum Meeting
  • Quick Meeting to touch base on:
    • What have I done? What am I doing next? What am I stuck on/need help?

• Sprint Retrospective
  • Review sprint process

• Sprint Review Meeting
  • Review Product
Planning

• Time estimation
• Tools
• Agile
• User stories
User Stories
User Stories

- **card**: a brief, simple requirement statement from the perspective of the user
- **conversation**: a story is an invitation for a conversation
- **confirmation**: each story should have acceptance criteria
The conversation

• An open dialog between everyone working on the project and the client
• Split up Epic Stories if needed
The Card

As a < type of user >, I want < some goal > so that < some reason >.

Who (User)

This should describe a fairly detailed user. It is not sufficient to just say “user.” Strive towards something like “broke college student on a mobile device user.” When we express the who with more detail we are able to better empathize with that particular user, determine the best solution and uncover implicit needs.

What (Goal)

The goal or action the user intends to take.

Why (Benefit)

Expressing the benefit to the user is by far the most important in my experience. Some of the most creative and inexpensive solutions come from the developers and users understanding why they are building something.
The Confirmation

• A confirmation criteria that will show when the task is completed
• Could be automated or manual
Exercise
How to evaluate user study?

Follow the INVEST guidelines for good user stories!

- independent
- negotiable
- valuable
- estimable
- small
- testable
• Schedule in any order.
• Not overlapping in concept
• Not always possible
• Details to be negotiated during development
• Good Story captures the essence, not the details
This story needs to have value to someone (hopefully the customer)

Especially relevant to splitting up issues
• Helps keep the size small
• Ensure we negotiated correctly
• “Plans are nothing, planning is everything” -Dwight D. Eisenhower
• Fit on 3x5 card
• At most two person-weeks of work
• Too big == unable to estimate
• Ensures understanding of task
• We know when we can mark task “Done”
• Unable to test == do not understand
Planning

• Time estimation
• Tools
• Agile
• User stories
Further Reading


Teamwork (Student Teams)

More on teams in real projects in the course
Expectation

• Meet initially and then regularly
• Review team policy
• Divide work and integrate
• Establish a process
• **Set and document clear responsibilities and expectations**
  • Possible Roles: Coordinator, Scribe, Checker, Monitor
  • Rotate roles every assignment
• Every team member should understand the entire solution
Dealing with problems

• Openly report even minor team issues in individual part of the milestone report
• In-class discussions and case studies
• Additional material throughout semester
• We will attend one team meeting
Planning and In-Team Communication

• Asana, Trello, Microsoft Project, ...
• Github Wiki, Google docs, ...
• Email, Slack, Facebook groups, ...
Meet your teammates!