CS189A
Software Engineering: Concepts and Practices

Jianwen Su
Department of Computer Science
UC Santa Barbara

https://capstone.cs.ucsb.edu/cs189a/cs189a_sched.html
CS189A: Week 3

- **Practice:**
  - Vision statement: Due Wednesday Oct 13
  - Sprint 1: deadline Thursday Oct 21
  - Next lecture (Oct 18): 1-slide team reports (3 minutes)
    Project goal, 1st sprint goal
  - Later: Product Requirement Document (release 1)

- **Principles:**
  - Intro to software engineering
  - Scrum & sprint
2-Page Vision Statement

- PDF via email to TA
  - Project Title / Name (can change)
  - Team name, members names_emails
  - Team lead
  - What the project is about
    - What problem the project is solving
      (what is innovation, the science, and new core technical advance)?
    - Why the problem is important
    - How the problem is solved today (if it is)
  - Identify the outcome of the project
  - Define initial project milestones: specification, design, prototyping
  - How do you plan to articulate and design a solution
    - List the implementation platform and technologies you plan to use to develop the solution
    - Process model you will use to achieve the milestones
Four 2-week sprints:
- Oct 11-21 (PRD v1 – tools, technologies, design, terminology);
- Oct 21-Nov 4 (use cases/user studies, prototyping, PRD v1);
- Nov 4-18 (design, prototyping, testing, PRD v2);
- Nov 19-Dec 3 (prototype demo/pres prep, prototyping and testing)

Specify what the product will do
- Vision statement (due in 2 days)
- Product Requirements Document (PRD)
- Design tools, brainstorming, coding (tests and implementation)

Build and test an initial prototype
- Typically teams iterate on these activities until they converge to a working prototype!
Introduction to the PRD: the Product Requirements Document

- The official statement of what is required of the system developers
- Includes a specification of both user and system requirements

- Defines **WHAT the system should do, not HOW it should do it**
  - Design/Impl comes later; give engineers freedom in how to go about it

- Agile and extreme SWE processes express requirements as
  - **Use cases** – how a system will act
  - Or as scenarios called **user stories** (describe result/benefit of it)
  - We will discuss/practice these next week
Agile Requirements Specification

1. Define project specifics
2. Team goals and objectives
3. Background and strategic fit
4. Assumptions
5. User Stories or Use Cases
6. User Interaction and Design
7. Questions
8. What we’re NOT Doing

- Evolve the document over time, concurrently with development

Required reading:
https://www.atlassian.com/agile/requirements
PRDv1: Your **Living** Requirements Document: A Shared Google Doc (due in ~2 weeks)

- Authors, Team, Project Title
- **Intro** – including problem, innovation, science, core technical advance (2-3 pages)
  - Define project specifics, team goals/objectives, background, and assumptions
- **System architecture overview**
  - High level diagram (1 page)
  - User interaction and design (1+ page)
- **Requirements (functional and non-functional)**
  - User stories or use cases (links) → 10 for PRDv1 prioritized
  - Prototyping code, tests, metrics (5+ user stories): github commits/issues
- **System models**: contexts, sequences, behavioral/UML, state
- **Appendices**
  - Technologies employed
Software Process Activities

1. **Software specification**
   - Customers and engineers define the software that is to be produced and any constraints/requirements on its operation

2. **Software design**
   - Software spec is designed and prototyped

3. **Software implementation, validation, and testing**
   - Software is programmed and checked to ensure that it is what the customer requires

4. **Software maintenance and evolution**
   - Software is maintained (bug fixes, upgrades) and modified to reflect changing customer and market requirements
Software Process Models

- Stages of software engineering: requirements specification, design, implementation, testing, maintenance

- Software process (software life-cycle) models
  - Determine the stages (and their order)
  - Establish the transition criteria for progressing from one stage to the next

- Software process models answer the questions:
  - What shall we do next?
  - How long shall we continue to do it?

- Models we’ll discuss: waterfall, spiral, evolutionary: agile/extreme
  - Waterfall (70s, 80s) when all software was “shrink wrapped and shipped”
  - Spiral (late 80s) risk-driven and iterative; Rational Unified Process (UP or RUP)
  - Evolutionary (late 90s, early 00s) as SW becomes increasingly online
Software is built iteratively and incrementally by first providing an initial version and then improving/ extending it based on the user feedback until an adequate system has been developed (late 90s, early 00s origin)

- Agile software development, extreme programming
- Triggered by change in application type (consumer, phones, web, cloud)

All activities are executed concurrently with fast feedback among them

Specifics impacted by application domain and deployment strategy (e.g. cloud/SaaS, web app)
Manifesto for Agile Software Development (2001)
available at: http://agilemanifesto.org/

“We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- **Individuals and interactions** over **processes and tools**
- **Working software** over **comprehensive documentation**
- **Customer collaboration** over **contract negotiation**
- **Responding to change** over **following a plan**

That is, while there is value in the items on the right, we value the items on the left more”
Extreme Programming (XP)

XP is a type of agile software development process proposed by Kent Beck (~late 90’s)
Extreme Programming (XP)

- Software is built iteratively, with frequent releases
  - Each release is a “working” system/product

- Each release implements the set of most valuable features/use-cases/stories that are chosen by the customer

- Each release is implemented in a series of iterations, each iteration adds more features/use-cases/stories

- Programmers turn the stories into smaller-grained tasks, which they individually accept responsibility for

- The programmer turns a task into a set of test cases that will demonstrate that the task is finished

- Working as pairs, the programmers make the test cases run, evolving the design in the meantime to maintain the simplest possible design for the system as a whole
**Scrum**

- An evolutionary/iterative/incremental/agile software process implementation
  - See: *Scrum and XP from the Trenches* -- free online book by Kniberg
Scrum

- An evolutionary/iterative/incremental/agile software process implementation
  - See: Scrum and XP from the Trenches -- free online book by Kniberg

- The main roles in Scrum are:
  - Scrum team: Team of software developers
  - Scrum master: Project manager
  - Product owner: Client

- Characteristics of Scrum:
  - Self-organizing teams
  - Product development in two to four week sprints
  - Requirements are captured as items in a list of product backlog
    - Yours will come from your requirements document (PRD)

- Homework: read the links on webpage under today’s date
Product Backlog (Scrum Artifacts)

An ordered list of everything known to be needed:
- These are the requirements – in your requirements document (PRD)
- A list of all desired work on the project
- Features, functions, requirements, enhancements, and fixes that constitute the changes to be made to the product in future releases

Prioritized by the product owner
- Reprioritized at the start of each sprint

Each backlog item also has an estimated time it will take to complete it
- Sum of tasks that make up an item (story, use case) should be the total

Product Backlog is never complete
Sprint and Scrum: Implementation

- **Sprint planning**
  - Use cases or user stories broken down into **tasks** (from prioritized product backlog) -> sprint backlog
    - Team members assign timings (how long each will take)
    - And pick tasks
  - Tasks = designing, implementing, testing, and demo'ing
    - Includes code review with a second member (demo individual task)
  - Any new tasks identified put onto backlog for inclusion next time

- **Daily standup**
  - What I did yesterday, what I'm doing today, + any blockers
  - Short/quick so done standing up!

- **At end of each iteration/sprint**
  - Sprint review: demo each story and end-to-end system to team
  - Retrospective and end of each iteration– what worked and what didn't
    - Vote on what to fix in the next sprint
Scrum Roles

- **Product owner (mentor + team in our case)**
  - Defines the features of the product
  - Decides on release date and content
  - Prioritize features according to market value
  - Adjust features and priority every iteration as needed
  - Accepts or rejects work results

- **Scrum Master (team lead in our case)**
  - Represents management of the project
  - Responsible for following the Scrum process
  - Ensures that the team is fully functional and productive
  - Shields the team from external influences
Scrum Roles

- **Scrum Team**
  - Typically 5 to 9 people
  - Cross-functional team that does the software development including designing, programming and testing
  - Co-location and verbal communication among team members
  - Teams are self-organizing, no titles
  - Team membership should not change during a sprint
Scrum Meetings

- **Sprint Planning**
  - This is done at the beginning of every sprint cycle (2 to 4 weeks)
  - Team selects items from the product backlog they can commit to completing
  - Sprint backlog is created
    - Tasks for this sprint are identified and each is estimated (hours, points, partial days). This is done collaboratively, **not** by Scrum Master
  - High-level design is discussed

- **Daily Scrum (at most 15 minutes)**
  - Daily, stand-up meeting
  - Not for problem solving
  - Every team member answers three questions:
    - What did you do yesterday?
    - What will you do today?
    - Is anything in your way? (Scrum Master is responsible for following up and resolving the impediments)
Scrum Meetings

- Sprint Review
  - Team presents what it accomplished during the sprint
    - Typically a demo of new features or underlying architecture
    - Incomplete work should not be demonstrated
  - Informal meeting, no slides
  - Whole team participates
  - Open to everybody
Scrum Meetings

- Sprint Retrospective
  - Periodically take a look at what is and is not working
  - Done after every sprint
  - Scrum Master, Product owner, Team and possibly customers and others can participate
  - One way of doing sprint retrospective is to ask everyone what they would like to
    1) Start doing,
    2) Stop doing,
    3) Continue doing
  Or
  1) What worked,
  2) What didn’t,
  3) What should change
Scrum Artifacts

- **Sprint Backlog**
  - Team members sign up for work (break stories into tasks) of their own choosing
  - Estimated work remaining is updated daily
  - Any team member can add, delete or change the sprint backlog
  - Each sprint backlog item has **daily estimates** for the amount of time that will be spent on that item each day

- **Burn Down Chart**
  - A daily updated chart displaying the remaining cumulative work on the sprint backlog. It gives a simple view of the sprint progress

- **Many tools on the web to track sprint**
  - Google Worksheet is easiest
  - Backlogs, burndown
  - Trello, PivotalTracker
Plan for Sprint 1

- Overview of project

- Sprint planning (using Trello): Backlog, on deck, in progress, done
  - Break up into tasks with durations (hours, part/days, points)
    - Identify/record initial requirements
    - Start writing sections of the PRD v1
    - Identify/install/test tools & technologies
    - Sketch out design and start listing terminology
    - Coding should be done for establishing basic use of technologies
      - Must become github commits this week
  - Assign 2+ members to each (implementer and tester/reviewer)
    - Fill 9 days according to durations for each member
    - Order tasks by priority (top = highest): top total 9 days * 5 members
  - Any new tasks identified put onto bottom of backlog for next time

- Setup burndown using google worksheet (shared w/ all)

- Daily standup/scrum (start tomorrow (scribe records in google doc))
Today (in Class)

- Work on vision statements (turn in as PDF via email to TA by Wednesday)
- Setup trello or pivotal tracker, go through tutorial
- Review technologies and getting started pointers
- Discuss and determine github workflow (read tutorials if needed)
  - Feature branch or gitflow: https://www.atlassian.com/git/workflows
  - Git branching basics
- Plan Sprint 1, include PRD v1 work (See previous years’ example projects)
  - Identify tutorials, technologies, configuration/deployment, prototyping (use of technologies)
    - Specify duration, review/testing step is to show another team member
  - PRD v1 work: intro (each member makes a pass), arch diagram, technologies
    - User stories/use cases (initial feature set) – more details on this next Monday
- Discussion section tomorrow:
  - First daily scrum starting tomorrow (set times for this week)
  - Daily (M-F) scrum time set (M and T in class/discussion)
  - Setup this week’s meeting with mentors if not done yet
Technology Ideas and Getting Started Pointers
More on Scrum

- More information about Scrum process is available at:
  - www.mountaingoatsoftware.com/scrum
  - www.scrumalliance.org
  - www.controlchaos.com

- Required reading
  - "Scrum/XP From the Trenches" by H. Kniberg. (Free with registration).