CAPSTONE LECTURE 1

Agile / Scrum

Background picture courtesy of Maree Reveley, used under Creative Commons Attribution-Share Alike 2.5 Generic license.

WHY A SOFTWARE DESIGN METHODOLOGY?

- In *ye olden days*, we thought hardware would be the hard part of computing
- Simple programmers would just implement the specs handed to them by the wise mathematicians and the powerful hardware designers
- They would just need to know how to type, really.

WHY A SOFTWARE DESIGN METHODOLOGY?

- We now know this is not the case, of course.
- We've learned that having some sort of structure allows us to develop software more effectively.
  - "Effectively" can mean different things in different domains and to different people!
    - An "effective" nuclear power controller will have very different benchmarks than an "effective" social media app.
    - Many different methodologies have been created which focus on different aspects of software development

WHY A SOFTWARE DESIGN METHODOLOGY?

- Different methodologies have different trade-offs. Examples:
  - Speed of Development vs Quality of Software
  - Documentation vs Dynamic Communication
  - Up-front Design vs Flexibility in Design
  - Feature Flexibility vs Planning
  - Adaptive Estimation vs Prescriptive Estimation
  - Risk-Averse vs Risk-Aware
  - Iterative vs Sequential

WHY A SOFTWARE DESIGN METHODOLOGY?

- Our methodology: Agile / Scrum
  - This is not the end-all, be-all of software development methodologies, but for the projects in this class, it fits well
  - Relatively lightweight
  - Flexible
  - Focus is on understanding customer needs

AGILE MANIFESTO

- http://agilemanifesto.org/
  - *Individuals and interactions* **over** *processes and tools*
  - *Working software* **over** *comprehensive documentation*
  - *Customer collaboration* **over** *contract negotiation*
  - *Responding to change* **over** *following a plan*

SCRUM

- Almost certainly the most popular agile methodology, although far from the only one…
  - Extreme Programming (XP)
  - Lean Software Development (LSD)
  - Dynamic Systems Development Method (DSDM)
  - Feature-Driven Development (FDD)
  - Agile Unified Process (AgileUP)
  - Crystal {Clear, Yellow, Orange, Red, Maroon}

SCRUM

- Not an acronym!
- Comes from a rugby scrum - everyone on team moving in one direction
- Teams are almost entirely self-managed
- Three roles
  - Product owner - Act as representative for the customer
  - Scrum master - They act as a “firewall” for the outside world and a centralized place to ask for help / facilitate meetings / etc.
  - Team - Everyone else: QA, developers, etc.

SCRUM

- Product-focused, end-user focused
  - Transparency - Work should be visible to those who need to see it
  - Inspection - Work should be examined regularly to ensure that the team is on the right path, or are doing things in a suboptimal manner
  - Adaptation - Work should be modifiable as requirements and limitations are better understood

USER STORIES

- A description in "plain language" that states what the user needs the software to do
- Related to requirements, but not exactly the same!
- Often in the *Connextra template*:
  - As a <role>
    I want <feature>
    So that <reason>

USER STORIES

- Examples:
  - As a manager
    I want to the software to display the current status of each engineer
    So that I can more effectively write status reports
  - As an Engineer
    I want the ability to enter my daily status on a web page
    So that I can update my manager on my status more easily
  - As a user of Excel
    I want a keyboard shortcut to select text
    So that I can quickly grab text without spending extra time reaching for my mouse

USER STORIES

- Allows us to not only see what they want, but more importantly, ***why***
- Gives us further flexibility if what they say they want is difficult/impossible, but can do something else that gives them the same result, or if there is a better way to achieve that objective

PRODUCT BACKLOG

- List of all items to be done
- In the beginning, should be all user stories
- Should be prioritized
- Differs from a Software Requirements Specification in that this is a living document - it will change as defects are added, user stories modified or removed, etc.
- Think of it as a kind of mixed to-do list / software specification

SPRINTS

- Software development is split into “sprints” - iterations of 2 - 3 weeks where work is done from the backlog (our sprints will be two weeks)
- At the beginning of the sprint, there is a sprint planning session where it’s determined which user stories will be worked on and who will work on which ones
- These are not set in stone! Some may run over or you may work on extra. There are various ways of estimating how much work can be done in a sprint (story points, velocity, etc.) but we will not use them for “our version” of Scrum
- This session is facilitated by the Scrum Master

SPRINTS

- At every point, and ESPECIALLY at the end of the sprint, you should have WORKING software
- It does not need to be feature-complete, but compiles, runs, etc.
- Adding a feature means it has met “the definition of done”
  - Code
  - Documentation
  - Integration
  - Testing

STANDUPS

- Standups - usually daily and very short communications with the rest of the team during the sprint
  - What have I done in the last 24 hours?
  - What do I plan to do in the next 24 hours?
  - Do I need any help or have any blockers?
- You can probably do this 3x/week, probably not necessary for every day
- However, this is up to you
- Facilitated by scrum master

RETROSPECTIVES

- At the end of each sprint, the team comes together to discuss:
  - What went well?
  - What could go better?
  - What can we do different in the next sprint?
- Once again facilitated by the scrum master

SPRINTS, STANDUP AND RETROSPECTIVES

- For our class, every other Friday (end of sprint), we will meet in class and have:
  - Retrospective on previous sprint
  - Sprint planning for next sprint
    - Before leaving, Scrum Master and I will discuss results of sprint planning
- Scrum Master position will change each sprint (different person each sprint).
  - Old scrum master handles retrospective; new scrum master handles sprint planning

WELCOME TO SPRINT 1

- You can't put together a backlog yet, but over the next two weeks:
  - Meet with the customer. I recommend face-to-face interaction.
  - Write up the needs of the customer in at least eight - ten user stories
  - Make basic decisions on software architecture (language, frameworks, tools, etc.)
  - Write up project proposal document and prepare "walking skeleton"

WELCOME TO SPRINT 1

- "Walking skeleton" - The basic "skeleton" of the software. A "Hello, world!" with test framework, basic code, etc. just to show that the basic system is set up on everybody's machines and that the software tools work (git, compiler, testing framework, etc)
- Please use GitHub or GitLab if possible
  - Make a private repository and add me (username: laboon on both services) as a collaborator
  - Have your walking skeleton program up and running by the end of the sprint (two weeks from today)
  - Proposal on how you will do the rest of the project, due at the end of the sprint (two weeks from today)

MOST IMPORTANTLY . . .

Have fun!