Agile Development and Project Management

CogSci 121 - HCI Programming Studio

Adapted from slides by Mountain Got Software, Shahid N. Shah
Agile Software Development

https://www.youtube.com/watch?v=OJflDE6OaSc
Manifesto for Agile Software Development

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.
The principles of agile methods

<table>
<thead>
<tr>
<th>Principle</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer involvement</td>
<td>Customers should be closely involved throughout the development process. Their role is provide and prioritize new system requirements and to evaluate the iterations of the system.</td>
</tr>
<tr>
<td>Incremental delivery</td>
<td>The software is developed in increments with the customer specifying the requirements to be included in each increment.</td>
</tr>
<tr>
<td>People not process</td>
<td>The skills of the development team should be recognized and exploited. Team members should be left to develop their own ways of working without prescriptive processes.</td>
</tr>
<tr>
<td>Embrace change</td>
<td>Expect the system requirements to change and so design the system to accommodate these changes.</td>
</tr>
<tr>
<td>Maintain simplicity</td>
<td>Focus on simplicity in both the software being developed and in the development process. Wherever possible, actively work to eliminate complexity from the system.</td>
</tr>
</tbody>
</table>
XP Practices

[See additional slides at the end]
XP: Embrace Change

- Recognize that:
  - All requirements will not be known at the beginning
  - Requirements will change
- Use tools to accommodate change as a natural process
- Do the simplest thing that could possibly work and refactor mercilessly
- Emphasize values and principles rather than process
Scrum in 100 words

- Scrum is an agile process that allows us to focus on delivering the highest business value in the shortest time.
- It allows us to rapidly and repeatedly inspect actual working software (every two weeks to one month).
- The business sets the priorities. Teams self-organize to determine the best way to deliver the highest priority features.
- Every two weeks to a month anyone can see real working software and decide to release it as is or continue to enhance it for another sprint.
Characteristics

• Self-organizing teams
• Product progresses in a series of month-long “sprints”
• Requirements are captured as items in a list of “product backlog”
• No specific engineering practices prescribed
• Uses generative rules to create an agile environment for delivering projects
• One of the “agile processes”
Scrum

Sprint goal

Sprint backlog

Product backlog

24 hours

Sprint
2-4 weeks

Potentially shippable product increment
Sprints

- Scrum projects make progress in a series of “sprints”
  - Analogous to Extreme Programming iterations
- Typical duration is 2–4 weeks or a calendar month at most
- A constant duration leads to a better rhythm
- Product is designed, coded, and tested during the sprint
Sequential vs. overlapping development

Rather than doing all of one thing at a time...

...Scrum teams do a little of everything all the time

Scrum framework

Roles
- Product owner
- ScrumMaster
- Team

Ceremonies
- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum meeting

Artifacts
- Product backlog
- Sprint backlog
- Burndown charts
Scrum framework

Roles
- Product owner
- ScrumMaster
- Team

Ceremonies
- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum meeting

Artifacts
- Product backlog
- Sprint backlog
- Burndown charts
Product owner

• Define the features of the product
• Decide on release date and content
• Be responsible for the profitability of the product (ROI)
• Prioritize features according to market value
• Adjust features and priority every iteration, as needed
• Accept or reject work results
The ScrumMaster

- Represents management to the project
- Responsible for enacting Scrum values and practices
- Removes impediments
- Ensure that the team is fully functional and productive
- Enable close cooperation across all roles and functions
- Shield the team from external interferences
The team

- Typically 5-9 people
- Cross-functional:
  - Programmers, testers, user experience designers, etc.
- Members should be full-time
  - May be exceptions (e.g., database administrator)
- Teams are self-organizing
  - Ideally, no titles but rarely a possibility
- Membership should change only between sprints
Scrum framework

Roles
- Product owner
- ScrumMaster
- Team

Ceremonies
- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum meeting

Artifacts
- Product backlog
- Sprint backlog
- Burndown charts
### Sprint planning meeting

#### Sprint prioritization
- Analyze and evaluate product backlog
- Select sprint goal

#### Sprint planning
- Decide how to achieve sprint goal (design)
- Create sprint backlog (tasks) from product backlog items (user stories / features)
- Estimate sprint backlog in hours
Sprint planning

- Team selects items from the product backlog they can commit to completing
- Sprint backlog is created
  - Tasks are identified and each is estimated (1-16 hours)
  - Collaboratively, not done alone by the ScrumMaster
- High-level design is considered

As a vacation planner, I want to see photos of the hotels.

- Code the middle tier (8 hours)
- Code the user interface (4)
- Write test fixtures (4)
- Code the foo class (6)
- Update performance tests (4)
The daily scrum

- Parameters
  - Daily
  - 15-minutes
  - Stand-up
- Not for problem solving
  - Whole world is invited
  - Only team members, ScrumMaster, product owner, can talk
- Helps avoid other unnecessary meetings
Everyone answers 3 questions

1. What did you do yesterday?
2. What will you do today?
3. Is anything in your way?

● These are commitments in front of peers
The sprint review

• Team presents what it accomplished during the sprint
• Typically takes the form of a demo of new features or underlying architecture
• Informal
  – 2-hour prep time rule
  – No slides
• Whole team participates
• Invite the world
Sprint retrospective

- Periodically take a look at what is and is not working
- Typically 15–30 minutes
- Done after every sprint
- Whole team participates
  - ScrumMaster
  - Product owner
  - Team
  - Possibly customers and others
Start / Stop / Continue

- Whole team gathers and discusses what they’d like to:

  - Start doing
  - Stop doing
  - Continue doing

This is just one of many ways to do a sprint retrospective.
Scrum framework

Roles
- Product owner
- ScrumMaster
- Team

Ceremonies
- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum meeting

Artifacts
- Product backlog
- Sprint backlog
- Burndown charts
Product backlog

- The requirements
- A list of all desired work on the project
- Ideally expressed such that each item has value to the users or customers of the product
- Prioritized by the product owner
- Reprioritized at the start of each sprint

This is the product backlog
### A sample product backlog

<table>
<thead>
<tr>
<th>Backlog item</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allow a guest to make a reservation</td>
<td>3</td>
</tr>
<tr>
<td>As a guest, I want to cancel a reservation.</td>
<td>5</td>
</tr>
<tr>
<td>As a guest, I want to change the dates of a reservation.</td>
<td>3</td>
</tr>
<tr>
<td>As a hotel employee, I can run RevPAR reports (revenue-per-available-room)</td>
<td>8</td>
</tr>
<tr>
<td>Improve exception handling</td>
<td>8</td>
</tr>
<tr>
<td>...</td>
<td>30</td>
</tr>
<tr>
<td>...</td>
<td>50</td>
</tr>
</tbody>
</table>
The sprint goal

- A short statement of what the work will be focused on during the sprint

**Database Application**

Make the application run on SQL Server in addition to Oracle.

**Financial services**

Support more technical indicators than company ABC with real-time, streaming data.

**Life Sciences**

Support features necessary for population genetics studies.
Attendance
(Agile) Project Management Tools

Gantt Chart: A bar chart. While visually appealing on a task/duration basis, it is limited because it does not show task or resource relationships well. Strength: easy to maintain and read.
(Agile) Project Management Tools

Network Diagram: A wire diagram, Also known as a PERT network diagram. A diagram that shows tasks and their relationships. Limited because it shows only task relationships. Strength: easy to read task relationships.
Project Control

Work with the client to determine the project needs & constraints (ANALYZE)
Define project milestones and deliverables (PLAN)

while project has not been completed or cancelled (EXECUTE)

  Draw up project schedule
  Initiate activities according to schedule
  Wait (for a while)
  Review project progress
  Revise estimates of project parameters
  Update the project schedule
  Re-negotiate project constraints and deliverables

if (problems arise) then
  Initiate technical review and possible revision
end if
end loop

Close project (DELIVER)
An example...

Writing a research paper
1: Requirements Definition

\{product goals\}

– 20 pages
– Double spaced
– On a topic addressing a question of the effectiveness of agile and waterfall methods
– Includes a literature review
– Includes a proposal for a research study
– Includes hypotheses & expected results
– IEEE citation format
– Reference at least 10 peer-reviewed papers
2: Work Breakdown Structure
{logical units of work to accomplish goals}

1. Planning
   A. Pick topic & research question
   B. Brainstorm potential research studies
   C. Make list of papers to read
   D. Document A-C in a proposal
   E. Discuss proposal with professor

2. Researching
   F. Read research papers
   G. Document key ideas

3. Writing
   H. Outline paper
   I. Write first draft
   J. Discuss draft with professor

4. Editing & Polishing
   K. Revise draft
   L. Check references and citation format
   M. Check length and formatting
   N. Proofread
   O. Submit paper
Project Management Tools

- Trello
- Basecamp
- Jira
- Asana
- Github + ZenHub
- Tom’s Planner
- Ganttter
- Github + Zenhub
Amy’s Personal Recommendation

**Trello**
For capturing requirements and sorting them into priorities

**Ganttter**
Turning requirements into a WBS and scheduling w/ dependencies

**Github+Zenhub**
Source control + feature tracking linked to commits *

*FREE cloud-based enterprise project scheduling tool*
Agile reading list

- *Agile and Iterative Development: A Manager’s Guide*, by Craig Larman
- *Agile Estimating and Planning*, by Mike Cohn
- *Agile Project Management with Scrum*, by Ken Schwaber
- *Agile Retrospectives*, by Esther Derby and Diana Larsen
- *Agile Software Development Ecosystems*, by Jim Highsmith
- *Agile Software Development with Scrum*, by Ken Schwaber and Mike Beedle
- *Scrum and The Enterprise*, by Ken Schwaber
- *Succeeding with Agile*, by Mike Cohn
- *User Stories Applied for Agile Software Development*, by Mike Cohn
Agile Development...
for your family

https://www.ted.com/talks/bruce_feiler_agile_programming_for_your_family?
Next Steps

• **Today:**
  - **Design Critique**, by Amy Fox
  - **Problems to Solve**, Jacob Browne

• **Thursday:** Design Critique, Elevator Pitch

• **Friday:** Technical Discussions / Studio (required)
  - Debriefing on Assignment 3
  - Quiz on Week 6 Content

• **Readings** (required)
  - Meirelles (*Design for information: an introduction to the histories, theories, and the best practices behind effective information visualizations*) —> Chapter 4 + Chapter 5

• **Next Week:**
  - *Design Critique: Data Flow and Architecture*
THANKS
Extreme programming

- Perhaps the best-known and most widely used agile method.
- Extreme Programming (XP) takes an ‘extreme’ approach to iterative development.
  - New versions may be built several times per day;
  - Increments are delivered to customers every 2 weeks;
  - All tests must be run for every build and the build is only accepted if tests run successfully.
Extreme Programming (XP)

• XP does not involve bungee cords!
  – It does not encourage blind hacking. It is a systematic methodology.
  – It predates Windows “XP” (2001)
• Developed by Kent Beck:
  – XP is “a light-weight methodology for small to medium-sized teams developing software in the face of vague or rapidly changing requirements.”
• Alternative to “heavy-weight” software development models (which tend to avoid change and customers)
  – “Extreme Programming turns the conventional software process sideways. Rather than planning, analyzing, and designing for the far-flung future, XP programmers do all of these activities a little at a time throughout development.”
  -- IEEE Computer, October 1999
Four Core Values of XP

- Communication
- Simplicity
- Feedback
- Courage
Communication

- XP emphasizes value of communication in many of its practices:
  - On-site customer, user stories, pair programming, collective ownership (popular with open source developers), daily standup meetings, etc.
- XP employs a coach whose job is noticing when people aren’t communicating and reintroduce them
Simplicity

• "Do the simplest thing that could possibly work" (DTSTTCPW) principle
  – Elsewhere known as KISS (Keep It Simple, Stupid!)

• A coach may say DTSTTCPW when he sees an XP developer doing something needlessly complicated

• YAGNI principle ("You ain’t gonna need it")
Feedback

• Feedback at different time scales
  • Unit tests tell programmers status of the system
  • When customers write new user stories, programmers estimate time required to deliver changes
  • Programmers produce new releases every 2-3 weeks for customers to review
Courage

- The courage to communicate and accept feedback
- The courage to throw code away (prototypes)
- The courage to refactor the architecture of a system

- Do you have what it takes?
The Extreme Programming Release Cycle

1. Select user stories for this release
2. Break down stories to tasks
3. Plan release
4. Evaluate system
5. Release software
6. Develop/integrate/test software
# Extreme Programming practices

<table>
<thead>
<tr>
<th>Principle or practice</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incremental planning</td>
<td>Requirements are recorded on story cards and the stories to be included in a release are determined by the time available and their relative priority. The developers break these stories into development ‘Tasks’</td>
</tr>
<tr>
<td>Small releases</td>
<td>The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the first release.</td>
</tr>
<tr>
<td>Simple design</td>
<td>Enough design is carried out to meet the current requirements and no more.</td>
</tr>
<tr>
<td>Test-first development</td>
<td>An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.</td>
</tr>
<tr>
<td>Refactoring</td>
<td>All developers are expected to refactor the code continuously as soon as possible code improvements are found. This keeps the code simple and maintainable.</td>
</tr>
</tbody>
</table>
# Extreme Programming practices

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pair programming</strong></td>
<td>Developers work in pairs, checking each other’s work and providing the support to always do a good job.</td>
</tr>
<tr>
<td><strong>Collective ownership</strong></td>
<td>The pairs of developers work on all areas of the system, so that no islands of expertise develop and all the developers take responsibility for all of the code. Anyone can change anything.</td>
</tr>
<tr>
<td><strong>Continuous integration</strong></td>
<td>As soon as the work on a task is complete, it is integrated into the whole system. After any such integration, all the unit tests in the system must pass.</td>
</tr>
<tr>
<td><strong>Sustainable pace</strong></td>
<td>Large amounts of overtime are not considered acceptable as the net effect is often to reduce code quality and medium term productivity</td>
</tr>
<tr>
<td><strong>On-site customer</strong></td>
<td>A representative of the end-user of the system (the customer) should be available full time for the use of the XP team. In an extreme programming process, the customer is a member of the development team and is responsible for bringing system requirements to the team for implementation.</td>
</tr>
</tbody>
</table>
Advantages of pair programming

- It supports the idea of collective ownership and responsibility for the system.
  - Individuals are not held responsible for problems with the code. Instead, the team has collective responsibility for resolving these problems.
- It acts as an informal review process because each line of code is looked at by at least two people.
- It helps support refactoring, which is a process of software improvement.
  - Where pair programming and collective ownership are used, others benefit immediately from the refactoring so they are likely to support the process.