Today

• Admin
• Teams: status?
• Topic Presentation: status?
• Questions
• Introduction to Ubicomp
• Agile Team Management
CSE 218 Presentations

Select a slot for your CSE 218 Presentation (2x students per slot)
- 2:00 PM - 2:15 PM slots are "Technology"
- 2:15 pm - 2:30 PM slots are "Research and Applications"

Thu Oct 08: Microsoft Kinect and Depth Cameras
Thu Oct 15: Google Glass and Augmented Reality
Thu Oct 22: Eye-Tracking and Accessibility
Thu Oct 29: Smart Homes and Smart Buildings
Thu Nov 05: Wearable Trackers
Thu Nov 19: Security and Privacy
Ubiquitous Computing

Definitions

Ubiquitous computing is the method of enhancing computer use by making many computers available throughout the physical environment, but making them effectively invisible to the user

– Mark Weiser

Ubiquitous computing, or calm technology, is a paradigm shift where technology becomes virtually invisible in our lives.

-- Marcia Riley
(Georgia Institute of Technology, Atlanta.)
Computing Everywhere

• many embedded, wearable, handheld devices communicate transparently to provide different services to the users

• devices mostly have low power and short-range wireless communication capabilities

• devices utilize multiple on-board sensors to gather information about surrounding environments
Experiment at PARC – TAB
Experiment at PARC - PAD
Experiment at PARC – BOARD
Transparent Interfaces

- hide their presence from user
- Provide interaction between user and application

Examples:
➢ Gesture recognition
➢ speech recognition
➢ free form pen interaction
➢ computational perception etc.
Interacting with Devices
Transparent Interfaces

**BUT**

Keyboard and mouse are still the most commonly used interfaces!!

**Need:**
- flexible interfaces
- Varied interfaces that can provide similar functionality
SketchPad
Ivan Sutherland - 1963
Novel devices
Novel devices

1. Foot controls
2. Eye-tracking
3. Multiple-degrees-of-freedom devices
4. DataGlove
5. Haptic feedback
6. Bimanual input
7. Tangible user interfaces
8. Handheld devices
9. Digital pens
10. Table top touch screens
11. Game controllers
12. Depth-Cameras
Context-Awareness

Context – information about the environment with which the application is associated.

LOCATION and TIME are simple examples of context!
Context-Awareness

Context aware application:
- is one which can capture the context
- assign meaning to it
- change behavior accordingly

Need:
Applications that are context aware and allow rapid personalization of their services.
Automated Capture

To capture our day-to-day experience and make it available for future use.

Constraints:
- Multiple streams of information
- Their time synchronization
- Their correlation and integration

Need:
Automated tools that support capture, integration and future access of info.
This is a beautiful and ingenious machine, very useful and convenient for anyone who takes pleasure in study, especially those who are indisposed and tormented by gout. For with this machine a man can see and turn through a large number of books without moving from one spot. Moreover, it has another fine convenience in that it occupies very little space in the place where it is set, as anyone of intelligence can clearly see from the drawing.

A. Ramelli. Le diverse et articiose macchine del Capitano Agostino Ramelli. Paris, 1588
V. Bush’s Memex

A memex is a device in which an individual stores his books, records and communications and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory.

Atlantic Monthly, "As we may think" July 1945
The Office of the Future

An in-depth analysis of how word processing will reshape the corporate office

... a relatively small but fast-growing group of companies will have moved into the office-of-the-future environment. The leap forward will be led by the “papermakers” – those companies that are involved primarily in generating, modifying, or moving paper.

These pioneers will have hooked together word-processing equipment into office systems to transfer information electronically and to move it into and out of central electronic files. For them, it will be the start of the paperless office

Paperless Office?

“The Office of the future

“I’ve finally achieved a paperless office.”
Digital Desk

Pierre Wellner, Xerox EuroParc 1992
Agile Project management
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>
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
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" (DTSTT TCPW) 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?
## 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

<table>
<thead>
<tr>
<th>Practice</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pair programming</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>Collective ownership</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>Continuous integration</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>Sustainable pace</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>On-site customer</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>
The Scrum process

Outline planning and architectural design

Assess

Select

Review

Develop

Project closure
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

24 hours

Sprint 2-4 weeks

Potentially shippable product increment

Return

Cancel

Gift wrap

Coupons

Product backlog

Coupons
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

Business conditions
- Team capacity
- Product backlog
- Current product
- Technology

Sprint goal

Sprint backlog
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.

**Life Sciences**

Support features necessary for population genetics studies.

**Financial services**

Support more technical indicators than company ABC with real-time, streaming data.
## A sprint backlog

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Mon</th>
<th>Tues</th>
<th>Wed</th>
<th>Thur</th>
<th>Fri</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code the user interface</td>
<td>8</td>
<td>4</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code the middle tier</td>
<td>16</td>
<td>12</td>
<td>10</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Test the middle tier</td>
<td>8</td>
<td>16</td>
<td>16</td>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>Write online help</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write the foo class</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Add error logging</td>
<td></td>
<td></td>
<td>8</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Tasks</td>
<td>Mon</td>
<td>Tues</td>
<td>Wed</td>
<td>Thur</td>
<td>Fri</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----</td>
<td>------</td>
<td>-----</td>
<td>------</td>
<td>-----</td>
</tr>
<tr>
<td>Code the user interface</td>
<td>8</td>
<td>4</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code the middle tier</td>
<td>16</td>
<td>12</td>
<td>10</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Test the middle tier</td>
<td>8</td>
<td>16</td>
<td>16</td>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>Write online help</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A Scrum 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
Next Steps

- Readings to discuss on Tuesday
  - V. Bush, "As We May Think", Atlantic Monthly, July 1945
Next Steps

- Read/Annotate both papers
- Submit your 1 pager summary
- URL will be communicated by Monday
- Deadline: Tue 2pm
- Next Thursday: Kinect and Depth Cameras presentation
Select a slot for your CSE 218 Presentation (2x students per slot)
- 2:00 PM - 2:15 PM slots are "Technology"
- 2:15 pm - 2:30 PM slots are "Research and Applications"

Thu Oct 08: Microsoft Kinect and Depth Cameras
Thu Oct 15: Google Glass and Augmented Reality
Thu Oct 22: Eye-Tracking and Accessibility
Thu Oct 29: Smart Homes and Smart Buildings
Thu Nov 05: Wearable Trackers
Thu Nov 19: Security and Privacy

2:00 PM: Thu 10/8/15
2:00 PM: Thu 10/15/15
2:00 PM: Thu 10/22/15
2:00 PM: Thu 10/29/15
2:00 PM: Thu 11/5/15
2:00 PM: Thu 11/12/15
2:00 PM: Thu 11/19/15
2:15 PM: Thu 10/15/15
2:15 PM: Thu 10/22/15
2:15 PM: Thu 10/29/15
2:15 PM: Thu 11/5/15
2:15 PM: Thu 11/12/15
2:15 PM: Thu 11/19/15
2:15 PM: Thu 11/19/15
Thanks