User Centered Design & Agile Integration

Mike Gerard
Contents

- Brief over of Agile development methodology
  - Rationale for Agile
  - Basic Agile concepts
  - Contrast with waterfall method
  - Roles on the Agile team

- Integration of User Experience (UX) / User Centered Design (UCD) into Agile
  - Coordinating UCD and development
  - UCD work products placed into Agile phases
  - Day-by-day examples for performing UCD in Agile cycles
  - Advantages and Challenges for UCD under Agile
Introduction to Agile

- We assume you’re already familiar with “waterfall” methods.
- We briefly cover Agile basics – key concepts and roles.
- We include links to more detailed Agile education and papers in an appendix.
Why Agile?

- The Agile methodology grew out of dissatisfaction with “waterfall” methods, for example:
  - Too much focus on documentation over working, useful code.
  - Slow
  - “Contractual” approach rather than frequent communication (early sign off on requirements followed by a period of development in a vacuum).
  - Unresponsive to new understanding, changing needs (requirements & design frozen – may match initial requirements but no what is needed by the customer at the time of deployment).
Waterfall versus Agile – High Level View

- In a waterfall method, it’s the overall system that is created across the phases – each phase occurs only once and the entire system is planned and designed “up front.”

- Under Agile, the “phases” are conducted repeatedly across cycles – each cycle is a “chunk” of functionality.

### Waterfall

- Analysis
- Design
- Coding
- Testing

#### Concept
#### Plan
#### Develop
#### Qualify

### Agile

Cycle n

- Analysis
- Design
- Coding
- Testing

Cycle n+1

- Analysis
- Design
- Coding
- Testing

Cycle n+2

- Analysis
- Design
- Coding
- Testing

Adapted from the Cutter Consortium’s "Agile Software Development" and User Centered Design and Agile.
How does Agile compare to waterfall methods?

- Agile assumes that everything is not known early on and expects a great deal of discovery throughout the process.
- Agile focuses on delivering working code to customers at frequent intervals.
  - The intended system is divided into discrete “bite size” functions, with the functions planned across development “cycles” (with each cycle lasting a few weeks)
  - Each cycle includes activities that would be spread across phases in a waterfall approach: analysis, design, coding and testing:

  ![Cycle Diagram](image)

- Change is “embraced” and expected – rather than “freezing” the requirements at an early stage, it is accepted that code developed early may need to change later.
- A small team collaborates constantly in each cycle - daily meetings involving all members of the Agile development team.
Roles on the Agile development team

- **Development**
  - Writes the code, sometimes uses “paired programming.”

- **Customer**
  - Articulates and clarifies the business process, requirements, etc.

- **UX/UCD practitioner**
  - Designs & evaluates the user interface, interaction model, etc.

- **“Scrum master”**
  - Similar to a project manager; leads and maintains the process.

*Other roles may be added as needed (e.g., a data modeler).*
Where is the end user in the process?

- **The key concept of User Centered Design is the inclusion of real end users throughout the development process.**
  - The advantages of including end users are not replaced by using Agile.
  - The same, well-accepted arguments for including end users in a waterfall process are applicable to Agile processes.

- **A common source of confusion is the “customer” role.**
  - The myth: “We are inherently performing UCD because we have a ‘user’ on the Agile development team: the Customer.”
  - The fact: The “customer” is NOT the end user, just as stakeholders and SME’s are NOT the end user in a waterfall project.

- **Including the customer throughout the development process is important and valuable, but doing so does not address the need for real end users.**
  - Typical Agile development is *customer-centered*, but rarely *user-centered*. 
Dangers of “customer centered design”

- “Customer” or “Executive” centered design often results in an interface that may fit the business’ needs in a way that is unusable by the target audience.
  - Uses language not familiar to actual users.
  - Provides an onerous workflow that is error-prone or more time consuming than necessary.
  - Doesn’t consider the environment of the end user (Noisy? Multitasking required? Need to stop in the middle and finish later?)

- Users may not adopt the system or, if the system is required, will find ways to “short circuit” the business process (e.g., skip steps, provide “creative” data, etc.).

- User satisfaction and productivity will be low.
  - “We don’t get the appropriate tools to do our jobs!”
Incorporating User Centered Design into Agile

- Early books, papers and web sites on Agile often neglected the critical role of a user interaction (or usability) team member.
  - This should not be read as an indication that those roles are not important; roles such as “data modeler” typically aren’t including in Agile descriptions.
  - Just as a complex data model will require special skills, an interactive user interface will require other skills.
  - This has changed in the last 2-3 years (some lessons have to be re-learned).

- There is a misconception that UCD is necessarily “waterfall” in nature, but the reality is that UCD already involves frequent and early evaluation with users.
  - Within Agile, those activities are accelerated (fewer users per evaluation, but more evaluations) and involve smaller, more molecular parts of the system.

- Lack of UCD involvement leads to the same problems whether the development process is waterfall or Agile:
  - Error-prone interactions; users can’t perform the tasks correctly
  - Inefficient workflow; users take longer than necessary to complete their tasks.
  - Lower adoption rates; users find ways to avoid using the system because it is too frustrating.
  - High help desk usage; calls regarding ‘simple’ tasks (can’t find a button, etc.).
  - Poor user satisfaction; many complaints, low morale.
Two key points for successful integration

1. **UCD is involved early.**
   - This means during “discovery” or “concept” or “cycle 0” (when the system is being conceptualized).
   - This provides time to learn about the users, recruit a “panel” or “pool” of participants.
   - This helps to develop a more holistic view of the system; how does everything fit together at a high level?

2. **Design (not just code) is expected to be iterative.**
   - Iterations based on user feedback, not just customer input.
   - User stories/requirements allowed to evolve with the UI designs.
Moving from waterfall to Agile – what changes for UCD?

- **Under waterfall, the entire system is designed & evaluated as a whole:**
  - Conceptual model (“wire-frames”) are generated and evaluated with users in Concept phase; every requirement is covered.
  - UI prototypes and design specifications for every screen are created and evaluated with users in Plan phase.
  - Requirements & design are “frozen.”

- **The process performed this way can be very time consuming:**
  - Takes time and effort to design an entire system.
  - User evaluations are generally large-scale: 4-8 users per user group, 1-4 hours per user; user evaluations can take several weeks and are planned weeks or months in advance.
  - Incorporating user feedback can be difficult (usually involves a change control process that may de-prioritize usability issues).

- **Within Agile, the techniques are the same but the method differs:**
  - Design & user evaluation is performed with small “chunks” of function; e.g., prototype and evaluate one screen in a cycle.
  - User evaluation is faster – fewer participants and less time per participant in a given cycle (across cycles the number of users may be equivalent to a waterfall process). 1-3 users for 15-30 minutes each.
  - User feedback is quickly incorporated – the user story and interface evolve *during* the evaluation (test one user, then update the prototype then test the next user, etc.).
When does interface design & evaluation happen?

- Most descriptions of Agile assume that the developers will design the UI as they code it.
- This approach is not effective for producing good, usable interfaces.
  - User needs are generally not heavily weighted, if they are even known.
  - Design & evaluation involves different skills than coding.
- If everyone on the Agile team is to begin working on the function on Day 1 of Cycle N, slack time is added.
  - Design takes time (both for thinking and for doing).
  - Developers wait for UI to be designed?
- The solution to these issues, observed in actual Agile practice, is the key to successfully integrating UCD into Agile:
  - UCD works with and ahead of development
Agile viewed as two tracks: Development and Design (UCD)

- **UCD works with and *ahead of* development:**
  - Work with customer and business analyst to flesh out “requirement” into a user story or use case.
  - Design for the cycle that development codes next.
  - Provide update to entire team during meetings daily meetings.

From Lynn Miller, *User Centered Design and Agile*, and Desiree Sy *Journal of Usability Studies*
Working ahead of and with development

- **Working ahead:**
  - User research (e.g., understand user and usability requirements for future cycles).
    - Note: Not expected to have a full understanding but need time to start considering how the pieces fit together.
  - Design the function for the next cycle (generally non-functional prototypes – drawings, graphics, simple HTML, etc.)
  - Evaluate the UCD prototype with small numbers of users and modify the prototype based on their feedback.
  - Work with the customer to revise the user story where necessary (for example, “This needs to be a soft delete, not a hard delete”).

- **Working with:**
  - Consult with developers on the code being developed in the current cycle; may need to modify designs based on development issues – the design is not “frozen” when it is given to the developers.
  - Help to prioritize functions and changes based on user research.
  - Perform user evaluations on working code when it is available (and reasonable to do so).
  - Discuss what is being learned about the next cycle, seek input from development where needed (e.g., development input on the UCD prototype).
Support for “working ahead” of development

- **This approach is receiving wide support:**
  - In the blogosphere
  - At conferences
  - In the literature

- **It is borne out in practical experience both at IBM and at other companies:**
  - More churn is introduced if work is completed simultaneously.
  - Usability and user experience reduced in priority when UI isn’t worked on ahead of the development cycle.

- **Some examples that have used the “work ahead” approach:**

  **Within IBM, across lines of business:**
  - Learning@IBM Explorer
  - Request 2 Contract (R2C)
  - Maintenance Contract Lifecycle Management (MCLM)
  - Field Resource System (FRS)
  - Rational Asset Manager (RAM), Rational
  - BlueHouse (hosted collaboration services for SMB)
  - SWG
  - Data Studio Administration Console
  - FPA
  - Rational Asset Manager 7.0 (first release)

  **Non-IBM**
  - eBay/PayPal
  - Delta Point of Sale applications
  - HP Self-Healing application
  - Mutual of Omaha
Agile UCD and parallel tracks – 1000 foot view

- UCD collaborates closely with both the development team and customer.
  - Important to note that UCD does **not** replace the “customer” role.
- UCD (with the customer) works slightly ahead of development to refine user stories and UI prototypes.

<table>
<thead>
<tr>
<th>Role</th>
<th>Cycle 0</th>
<th>Cycle 1</th>
<th>Cycle 2</th>
<th>Cycle 3</th>
<th>Cycle 4</th>
</tr>
</thead>
</table>
| Develop activities | - High level design for all stories  
      | - Detail technical requirements  | - Detail design for cycle 2 stories 
      | - Implement non UI code  | - Detail design for cycle 3 stories  
      | - Implement and test cycle 2 stories  | - Detail design for cycle 4 stories  
      | - Implement and test cycle 4 stories  |                      |
| Usability activities | - User research  
      | - Conceptual model  
      | - Prototype cycle 2 stories  | - Test cycle 2 designs  
      | - Prototype cycle 3 stories  | - Test cycle 3 designs  
      | - Prototype cycle 4 stories  | - Test cycle 4 designs  
      | - Prototype cycle 5 stories  | - Test cycle 5 designs  
      | - Prototype cycle 6 stories  |                      |
| Customer activities | - Outline all stories  
      | - Detail stories for cycle 1  | - Detail stories for cycle 2  
      | - Refine stories for cycle 1 when asked  | - Detail stories for cycle 3  
      | - Refine stories for cycle 2 when asked  | - Detail stories for cycle 4  
      | - Refine stories for cycle 3 when asked  | - Detail stories for cycle 5  
      | - Refine stories for cycle 4 when asked  |                      |
Where do UCD activities fit into Agile?

### Development activities

<table>
<thead>
<tr>
<th>Cycle 0</th>
<th>Cycle 1</th>
<th>Cycle 2</th>
<th>Cycle n</th>
</tr>
</thead>
</table>
| **Analyze**  
  - High level code design  
  - Technical requirements | **Analyze**  
  - User stories with team  
  - Technical requirements | **Analyze**  
  - User stories  
  - Test Code  
  - Design Code  
  - Code User story function | **Analyze**  
  - User stories  
  - Test Code  
  - Design Code  
  - Code User story function |

### UCD activities

<table>
<thead>
<tr>
<th>Cycle 0 (or Concept phase)</th>
<th>Cycle 1</th>
<th>Cycle 2</th>
<th>Cycle n</th>
</tr>
</thead>
</table>
| **Analyze**  
  - SOW  
  - UDEP | **Analyze**  
  - User stories with team | **Analyze**  
  - User stories with team | **Analyze**  
  - User stories with team |
| **Design**  
  - Requirements  
  - Scenarios  
  - User profiles | **Design**  
  - User stories with team  
  - High level conceptual model | **Design**  
  - User stories with team  
  - UI prototype  
  - UI spec | **Design**  
  - User stories with team  
  - UI prototype  
  - UI spec |
| **Test**  
  - Heuristic review  
  - Competitive analysis | **Test**  
  - Accessibility review  
  - Walkthrough | **Test**  
  - Accessibility review  
  - Walkthrough  
  - Implementation review | **Test**  
  - Accessibility review  
  - Walkthrough  
  - Implementation review  
  - Usability test |
How can all of this fit into short Agile cycles?

- **Activities are performed on small “chunks” of function.**
  - Function being evaluated for a cycle may be a single dialog box or a few fields on an input form.

- **Prototypes should be simple and the minimum fidelity that will allow reasonable feedback.**
  - Typically UI prototypes of sufficient fidelity can be produced in a short time – an hour or two.
  - Unlike a waterfall approach, working code will be available for evaluation in just a few weeks, so lower fidelity is not as risky.

- **Evaluations performed on a smaller scale.**
  - Short sessions (~15-30 minutes) with small numbers of users (1-4).
  - Consider the RITE (Rapid Iterative Testing & Evaluation) method: Design, evaluate with one user, re-design, evaluate with another user, re-design, etc.

- **Documentation is reduced.**
  - Output = changes to a prototype and a short summary (e.g., 1 page of bulleted findings) rather than a large, formal report.
# Day-to-day UX activities in a typical Agile Cycle

*Note: This cycle does not include a usability test since the functionality is too limited.*

## Cycle 3

<table>
<thead>
<tr>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
</tr>
</thead>
</table>
| Cycle planning meeting  
  - Review user stories & design for Cycle 3  
  - Review requirement for Cycle 4  
  - Prioritize work queue  
  - UX begins design for Cycle 4  
  - UX & customer starts on user story for Cycle 5.  
  - Development starts coding for Cycle 3.  
| Daily status meeting  
  (entire team discusses progress, plans, impediments).  
  - UX designs for Cycle 4.  
  - UX collaborates with customer on user story for Cycle 5.  
  - UX collaborates with development on Cycle 3 (design consultation).  
| Daily status meeting  
  - UX designs for Cycle 4.  
  - UX collaborates with customer on user story for Cycle 5.  
  - UX collaborate with development on Cycle 3.  
| Daily status meeting  
  - UX designs for Cycle 4.  
  - UX collaborates with customer on user story for Cycle 5.  
  - UX collaborates with development on Cycle 3.  

**Week 1**

<table>
<thead>
<tr>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
</tr>
</thead>
</table>
| Daily status meeting  
  - Usability walkthrough on design for Cycle 4.  
  - Validate Cycle 5 user story with appropriate stakeholders.  
  - Collaborate with development on Cycle 3.  
| Daily status meeting  
  - Usability walkthrough on design for Cycle 4.  
  - Revise Cycle 5 user story.  
  - Collaborate with development on Cycle 3.  
| Daily status meeting  
  - Revise design for Cycle 4.  
  - Revise Cycle 5 user story.  
  - Collaborate with development on Cycle 3.  
| Daily status meeting  
  - Revise design for Cycle 4.  
  - Revise Cycle 5 user story.  
  - Collaborate with development on Cycle 4.  

**Week 2**

<table>
<thead>
<tr>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
</tr>
</thead>
</table>
| Daily status meeting  
  - Completed code for cycle 3.  
  - Completed design for Cycle 4.  
  - Completed user story for Cycle 5.  
  - Sigh of relief.  
| Reflection meeting  
  - Discuss progress, plans, and impediments  
  - What worked, what didn’t?  
  - Completed code for cycle 3.  
  - Completed design for Cycle 4.  
  - Completed user story for Cycle 5.  

---

**Note:** This cycle does not include a usability test since the functionality is too limited.
A typical 2-week cycle that includes testing

- **This cycle includes a usability test** – but keep in mind that:
  - Usability Tests are small scale (few users, few scenarios).
  - Instead of a long, formal report problems added to the backlog (a 1-2 page summary listing problems and recommendations).

### Cycle 4

<table>
<thead>
<tr>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
</tr>
</thead>
<tbody>
<tr>
<td>Week 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| - Cycle planning meeting  
  - Review user stories & design for Cycle 4  
  - Review requirement for Cycle 5  
  - Prioritize work queue  
  - UX begins design for Cycle 5  
  - UX & customer starts on user story for Cycle 6.  
  - Development starts coding for Cycle 4.  
| - Daily status meeting (entire team discusses progress, plans, impediments).  
  - UX designs for Cycle 5.  
  - UX collaborates with customer on user story for Cycle 6.  
  - UX collaborates with development on Cycle 4.  
  - **Usability test on Cycle 1-3 code.**  
| - Daily status meeting  
  - UX designs for Cycle 5.  
  - UX collaborates with customer on user story for Cycle 6.  
  - UX collaborate with development on Cycle 4.  
  - **Usability test on Cycle 1-3 code.**  
| - Daily status meeting  
  - UX designs for Cycle 5.  
  - UX collaborates with customer on user story for Cycle 6.  
  - UX collaborates with development on Cycle 4.  
  - **Usability test on Cycle 1-3 code.**  
| - **Daily status meeting**  
  - UX designs for Cycle 4.  
  - UX collaborates with customer on user story for Cycle 5.  
  - UX collaborate with development on Cycle 3.  
  - **Usability test on Cycle 1-3 code.**  
| - **Reflection meeting**  
  - Discuss progress, plans, and impediments  
  - What worked, what didn’t?  
  - Completed code for cycle 3.  
  - Completed design for Cycle 4.
  - Completed user story for Cycle 5.
  - **Problems found in usability test added to backlog / work queue.**  

| Week 2 |
| - Daily status meeting  
  - Usability walkthrough on design for Cycle 4.  
  - Validate Cycle 5 user story with appropriate stakeholders.  
  - Collaborate with development on Cycle 3.  
  - **Usability test on Cycle 1-3 code.**  
| - Daily status meeting  
  - Usability walkthrough on design for Cycle 4.  
  - Revise Cycle 5 user story.  
  - Collaborate with development on Cycle 3.  
  - **Usability test on Cycle 1-3 code.**  
| - Daily status meeting  
  - Revise design for Cycle 4.  
  - Revise Cycle 5 user story.  
  - Collaborate with development on Cycle 4.  
  - **Usability test on Cycle 1-3 code.**  
| - Daily status meeting  
  - Revise design for Cycle 4.  
  - Revise Cycle 5 user story.  
  - Collaborate with development on Cycle 4.  
  - **Usability test on Cycle 1-3 code.**  
| - Reflection meeting  
  - Discuss progress, plans, and impediments  
  - What worked, what didn’t?  
  - Completed code for cycle 3.  
  - Completed design for Cycle 4.
  - Completed user story for Cycle 5.
  - **Problems found in usability test added to backlog / work queue.**  

---
Advantages of an Agile approach for user experience

- **The intent of Agile is to be responsive and flexible:**
  - Less likely to hear, “Sorry, that’s not what the requirements say” when a usability problem is uncovered.

- **Functional software is available for user evaluation at frequent intervals.**
  - Rather than a single large test just prior to deployment that uncovers problems too late to do anything about them.

- **UCD is fully integrated into the team**
  - Alleviates some of the “disconnected” or “outsider” status that UCD sometimes encounters.
  - Communication is essential for integration – daily productive meetings under Agile.

- **Small team reduces the “diffusion of responsibility”**
  - Everyone feels “on the hook” for the user experience, everyone contributes.

- **Reduction in documentation**
  - Time saved by eliminating large reports - focus instead on improving the prototype or code.
Challenges for user experience

- **Can be very difficult to think holistically about the application – you may not have an overall “big picture” or conceptual model.**
  - For an existing application that is being updated, not a tremendous hurdle. For a project being built from scratch, this will be a major issue.
  - More important than ever to include UCD early (concept/discovery phase) in the project so that there is time to give thought to the holistic view of the system.

- **Need access to users very quickly and on a frequent basis.**
  - Tests are small scale, covering small bits of function but still takes time to find and schedule users.
  - Need to use your judgment about when it is appropriate to collect user feedback (e.g., do you need a walkthrough for adding a set of “Yes” and “No” action buttons?).
  - Create a pool of users during cycle 0; this will speed user evaluations later.

- **UX sizing versus development sizing can be very different, so aligning cycles will be a learning process.**
  - Need to make sure the UX work can be completed at the right pace – need more practitioners for some cycles and fewer for others?

- **Breakneck pace.**
  - Simultaneously working on building requirements, designing a UI, and evaluating the UI is mentally taxing.

- **Although co-location is not essential, disparate time zones can be even more challenging.**
  - Being able to get on the phone or ask a question via instant messaging is even more important in an Agile setting; a day’s wait for an answer can feel like a long time when you’re dealing with a 10 day cycle.
Professional Organizations

Human Factors and Ergonomics Society
Usability Professionals Association
Computer-Human Interaction (ACM Special Interest Group)