UI Design and Agile Development
Building Great Interfaces with Incremental Design

Brian Zeichick
TeamForge Product Manager
Overview - UI Design And Agile Development

• What is Agile
  • History of Agile & Common Types of Agile Methodologies
• Agile Myths
  • Common misconceptions of agile
• Mistakes to Avoid
  • Design all interfaces up-front & handing over design to development
• Getting Started
  • Changing your Mindset & stop creating Hi-Fi Prototypes
• In Practice
  • Working with the team, communicating with users
  • Incremental/Iterative Design via low-fi mockups and prototypes
What is Agile?

- Development process that focuses on iterative development
- Requirements and solutions evolve and adapt over time
- Facilitated through collaboration of self-organized teams
- The term “Agile” was coined in 2001 for the Agile Manifesto
  - Core Values Included:
    - Individuals and interactions over processes and tools
    - Working software over comprehensive documentation
    - Responding to change over following a plan
- Agile is a software development process that:
  - Has shorter planned cycles
    - Smaller set of requirements identified more frequently
  - Frequent software delivery
    - Focus on “working” software for every iteration
  - Closer Customer Relationship via:
    - Constant Feedback loop
    - Frequent User Acceptance Testing
Different Agile Methodologies

- **Lean Software Development**
  - Organization or Division level process
  - First described by Mary & Tom Poppendieck – *Lean Software Development*
    - Eliminate Waste
    - Decide as late as possible
    - Empower the team
    - See the whole
    - Amplify Learning
    - Deliver as fast as possible
    - Build Integrity In

- **Scrum**
  - Team Level process designed to increase speed & flexibility in product development
  - Most popular Agile Methodology
  - Daily Standups, Retrospectives, and Planning Meetings
  - Backlogs and burndowns
  - ‘Pig’ Roles – Product Owner, Scrum Master, and the development team
  - ‘Chicken’ roles – Stakeholders (customers, vendors), Product Executive, Managers

- **Extreme Programming**
  - No, it doesn't involve Bungee cords or parachutes
  - Created by Kent Beck while working for Chrysler
  - Individual Contributor Level Agile Methodology
  - Pair Programming
  - Extensive Code Reviews
  - Unit Testing via Continuous Integration
Agile Myths – Bad for User Experience

- It takes time to design a great interface, before development
- With such a short development cycle, UI can’t fit in
- If you don’t do Usability Tests before implementation, you’ll be left with a poor design that will be very hard to change later
- Product Design becomes a bunch of ad-hoc tactical decisions
- Little time is left for contextual inquiries - the product becomes the "company’s" vision rather than the consumer’s asked-for solution
- “Featuritis” is a full-blown epidemic
- Redesign of the entire system is needed every time a new feature changes for how the ones already in place interact
- Starting development soon leaves little time for Concepting, UX research and designing a vision.
- Causes Brand Damage – when releasing too early the brand suffers
Agile is actually Good for User Experience

• Jakob Nielsen has recently done research about the combination of UX and Agile.
• According to their study the team members (ux people and developers) were very satisfied with the results: both the level of integration of UX into the process as the satisfaction with the Agile method were very high compared to the Waterfall method
• The study asked participants how extensively UX was integrated into their projects and how satisfied they were working on projects with a particular development methodology. They indicated their answers on a 1–5 scale, with 5 indicating the highest level of integration or satisfaction:

<table>
<thead>
<tr>
<th>Project Methodology</th>
<th>Integration of User Experience</th>
<th>Satisfaction with the Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Waterfall</td>
<td>2.5</td>
<td>2.9</td>
</tr>
<tr>
<td>Agile</td>
<td>3.1</td>
<td>3.7</td>
</tr>
<tr>
<td>Iterative</td>
<td>3.2</td>
<td>3.8</td>
</tr>
</tbody>
</table>

Clearly, Agile is considerably better than the old Waterfall method.
Other Agile Myths

• Burns out Team
  • You end up working hard all through the product lifecycle

• Misses Target of Initial Requirements
  • Designers work-out all of the big decisions in advance
  • Designers think large-scale and can avoid tactical decisions

• You can’t do a project that has complex requirements using Agile
  • Never use agile if you need “Rocket Science”
Agile and Rockets?!

- Starting when Scaled Composites made kits for planes
  - Before this approach all homebuilt airplanes had a complex blueprint
    - Sometimes there was an accompanying book of instructions
  - Based on Simplicity Dress Patterns
    - Series of sketches with a few steps for each sketch
    - Idea was to break down each step simply so that it would be bullet-proof
    - Sent out new set of plans in Quarterly Newsletter
    - When people reported issues fixes would be in the next newsletter
    - This process lasted for ~30 years

- Going to Space in a Surfboard
  - SpaceShipOne won the Ansari X Prize in 2004
  - Fostered “Do the simplest thing first” mantra
    - This led to uncovering solutions even when the design didn’t work
  - Stayed one step ahead of the Engineers with CAD drawings of what was being built next
  - Breakthrough aerodynamics – “Feathered Re-Entry”
  - SpaceShipOne could never have been created using a “traditional” aerospace development approach
Mistakes to Avoid – Design All Interfaces Up-front

• Leads to excessive designs that often get thrown-away or redesigned
• The foundation of the design is based on a loose understanding and is destined to change
• Customer needs and requirements change frequently, so should the user interface designs
• It’s impossible to think through all issues initially
• Technology improvements and changes often require UI rework
• Any new feature imposes constraints on what can be done in the future, so an unnecessary feature now may prevent implementing a necessary feature later
• Don’t over-engineer something based on what you think you might need later on.
Mistakes to Avoid – Handover Design to Development

- Agile doesn’t equate to giving up and handing over design completely
- Somebody on the team has to have the determination to ensure that the design quality stays high
- Programming can happen without design, but it’s up to you to get involved
- Resist the temptation to call a design “Good Enough” so that it fits in a sprint
- Sacrifice Good UI for Quick Dev
Getting Started – Change Your Mindset

• Keep big picture in mind
• Make small course corrections
• Think about what can be done right now
• Lose the mantra, “Wouldn’t it be great, if…”
• Concentrate on standards rather than wiz-bang UI
• Think of design as evolutionary throughout the process, rather than a plan that you need to stick to
• Own the User Experience portion of the product
• Work closely with the Product Owner and development team to flesh out stories and mockups
Getting Started – Stop Creating Hi-Fi Prototypes

• Stop Creating Hi-Fi Prototypes with external tools
  • Axure & iRise
• Breadth over Depth
  • Make each design work a little bit
• Avoid designing each feature down to the finest detail
  • Reduces throw-away work
• Instead work closely with developers to create working code that can be demonstrated to stakeholders and customers
In Practice – Establish a Team Rhythm

• Daily Standups are your opportunity to sync-up with developers
• Create necessary follow-ups with developers
  • Make course corrections as needed
  • Don’t wait for UAT or Sprint Review to give feedback
• Stay ahead of the team (but not too far)
  • Having a design ready for a developer is important to keep things moving
  • Keeping ahead will allow you to still design useful interactions
In Practice – Open communication with Users

• User Acceptance Testing should be done as often as possible
  • Ideally when each feature is demonstrable, or at the end of every sprint
  • Ensure stakeholders and relevant team-members are there
    • Product Owner, Scrum Master, and developers at least

• Keep an open dialog with users in the most time and budget-efficient way
  • No need for an expensive usability lab. It’s best for users to be in their own working environment
  • Try to have several shorter interactions with users rather fewer longer interactions

• Adapt to Changing Customer Needs
In Practice – Incremental/Iterative Design

- Developed in response to weakness of Waterfall process
- Incremental/Iterative Design is a hierarchical design process that:
  - Resuses results for unchanging portions of the design
  - Limits changes to smaller portions of the design
  - Timing constraints on the design should be realistic
- Break design into logical groups
  - Ideally should mirror User Stories
  - Should be able to be implemented separately from other designs
- Embrace “You Ain’t Gonna Need It” (YAGNI)
  - Reduces throw-away designs
  - Helps focus on what’s really needed
- Keep it Simple
  - Complexity does not equal better design
  - Taking simple approach unveils possible optimizations
- Refactoring is key
  - Allows for quick changes to be made
  - Makes iterative design possible
In the 1970s, many countries decided to invest in wind energy.

Denmark, a small country of around 5 million people, became a world leader in this technology using an incremental approach while more formal design processes in the US, Germany and the United Kingdom failed to develop competitive machines.

The reason for the difference of approach was that the Danish wind industry developed from an agricultural base whilst the American and UK wind industries were based on hi-tech aerospace companies with significant university involvement.

While the Danes built better and better windmills using an incremental approach, those using formal planning techniques believed that they could easily design a superior windmill straight away.
In Practice – Mockups

• Long-Term Vision and Short-Term Goals
• Keep it Simple – Do the simplest thing that could possibly work
• Sometimes it makes sense to compromise on a design
• Try to think of Non-Goals or “You Ain’t Gonna Need It”s
• Be willing to do something that requires less dev
• Sometimes doing the simplest thing first uncovers other solutions that would have been hard to find if you started with a more complex design
• Build in flexibility into the design so that it can change easily when requirements change
In Practice – Low-Fidelity Mockups

• Think of it as a conversation starter, rather than a design mandate
• Paper & Pencil or Whiteboard (screenshot)
• Balsamiq, MockingBird, Mockflow
• Make it accesible to your team
• Integrate mockups into user stories and other PBIs
  • Attach to wikis, work items, and artifacts
• Low-Fidelity alludes to design is negotiable, and is not set in stone
In Practice – Creating Spike Implementations

• Sometimes in order to get the design right, you need to create a Spike or Analysis Story
  • A Spike is a quick story that focuses on a piece of the requirement
  • Important if it’s not clear on what the right direction to take
• Often needed if it seems like a great design is crucial for the feature’s success
In Practice – Ensure Design Standards are in Place

- Ensures Consistency
- Reduces UI as a bottleneck
- A good knowledge of patterns: not just the solutions but also appreciating when to use them and how to evolve into them
In Practice – Get Familiar with UI Development

• Familiarize yourself with implementation technologies
  • HTML, Javascript, AJAX libraries (Dojo, YUI, JQuery, GWT) for web projects
  • Swing Designer or Spring for java
  • Visual Studio for C# and .net
  • Flash, Silverlight, Flex for RIAs
• Helps create Productizable Prototypes
• Regardless of process, this helps understanding the overall feasibility of the design
• Develop Refactoring skills so you can confidently make improvements whenever you see the need
In Practice – Create Productizable Prototypes

• Create prototypes that can be rolled into the product, rather than throw-away ones
• Strive to create prototypes that can test out several ideas
• Build in flexible options to flesh out design questions
• Embrace the Technology that will be used in the final implementation of the product
  • If you’re not expert, pair up with one
• Have a constant desire to keep code as clear and simple as possible
Summary - UI Design And Agile Development

• What is Agile
  • History of Agile
  • Common Types of Agile Methodologies
• Agile Myths
  • Common misconceptions of agile
• Mistakes to Avoid
  • Design all interfaces up-front
  • Hand-over design to development
• Getting Started
  • Change your Mindset
  • Stop Creating Hi-Fi Prototypes
• In Practice
  • Establish a team Rhythm
  • Open Communication with Users
  • Incremental/Iterative Design
  • Low-Fidelity Mockups
  • Ensure Design Standards are in place
  • Familiarize yourself with UI Development
  • Create Productizable prototypes
Any Questions or comments?

Thank You for attending.