Agile UX method adaptation
Skills to successfully put user-centered design into agile projects
Desirée Sy & John Schrag
CHI 2012, May 2012

Tutorial leaders

John Schrag
john.schrag@autodesk.com
@JVSchrag

Desirée Sy
desiree.sy@autodesk.com
@DesireeSy
Course context
Where we’re starting from, where we’re going to

Baseline

• Agenda
• Who’s here?
• Agile terminology
Agile UX toolkit
(day-long tutorial)

morning

afternoon
Agile UX planning

Agile UX tactics

optimal process

managing reality
managing reality

Agile UX tactics

Parallel track workflow  Timing of agile UX activities
Day to Day  Timing of communication
BREAK
Staged stories  Incremental implementation
Design chunking  Incremental design
Who are we?

Alias, now Autodesk:

- Commercial, shrink-wrapped software
- 3D graphics, highly interactive
- Non-standard innovative UI
- Users: Creative professionals
- Tasks: Generative, open-ended, complex

“Adapting Usability Investigations for Agile User-Centered Design”

“Case Study of Customer Input For a Successful Product”
OH: "Do you want a pony, or four pieces of pony that don't fit together?"

2:01 PM Jun 8th via web
Retweeted by 1 person

unixronin
Erik Fichtner

We ❤ agile UX
Agile UX: the good

- “Done” includes design
- Issues get fixed
- Less “design drift”
- Less wasted design
- Face-to-face is better than “over the wall”
- User data has effect on current release
- Most important features are done first
- Contextual inquiry & usability testing on actual product
- Satisfying to see designs in real use
- Enables requirements iteration

Who’s here?

- Practitioner/academia?
- For practitioners: what product/service?
- For practitioners: UX maturity of practice?
- Who has had agile training?
- Roles?
- Who’s working agile now?
Baseline

• Agenda
• Agile terminology
• Transitioning problems

Agile manifesto

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.

http://agilemanifesto.org/
Agile vs. waterfall

“Waterfall”

Analysis → Design → Coding → Testing

Agile

Analysis → Design → Coding → Testing

Analysis → Design → Coding → Testing

- Adapted from Cutter Consortium: “Agile Software Development”

agile qualities

iterative
incremental
continuous
collaborative
Agile development cycles

Miscellaneous terms

Iteration/Cycle/Sprint/Stage (sorry!)
Scrum master
Feature cards/Tasks
Retrospectives
Feature Points/Story Points
Velocity
Working version
Preview release vs. Product release
Some story myths

• Story = feature
• Story = specification
• Story must fit in one iteration
• All stories have firm estimates and specs in Iteration Zero (or even Iteration One)

These are NOT true.
“Story”

= 

User Problem

with acceptance criteria

“Cultural change isn’t easy. You need time, patience, & iteration.”
Parallel-track workflow
a.k.a. Staggered sprints

agile qualities
iterative
incremental
continuous
collaborative
Agile design timing: Parallel tracks

Developer track: Focus is on production code
Interaction designers track: Focus is on user contact

Iteration 1: Developer track

Underlying architecture work
Critical features with little user interface design required
Iteration 1: Interaction designers

Design, create prototypes, usability test, and iterate (RITE method)
Field studies to understand user needs (contextual inquiry)

Iteration 2: Developers

Take the verified designs and start making them a reality
Iteration 2: Interaction designers

Usability test completed code for integration and implementation issues

Use data gathered in the last iteration to create designs for next iteration
Iteration 2: Interaction designers

Usability test completed code for integration and implementation issues
Use data gathered in the last iteration to create designs for next iteration
Field studies for detailed information needed for upcoming iterations

And so on...

Constant communication between the two tracks is essential for success
These are not just hand-offs
**Discussion:** Parallel track timing

- Who is using this?
- How is it working?
- Questions, or problems?

“To fit into an agile development process, usability testing must also become agile”
Adjusting usability testing

Agile usability testing should:
• Enable iterative improvement
• Have very fast turnaround
• Be time-boxed and occur regularly
• Value in-product change over documents
• Engage the whole team

Formative vs. Summative

• All of our usability testing is now formative
• Protocol includes design goals to achieve
• Test as early as possible
• Iterate on prototypes between testers
• The focus has moved from finding problems to validating solutions
Agile usability testing

• Use RITE testing (see reference)
• Usability test at regular intervals
  (book first, fill out protocols just-in-time)
• Recruit continuously
  (for agile, you can re-use testers)
• Use user proxies, remote testing, beta testers
• See also Design Chunking (later)

Further references

• RITE (Rapid Iterative Testing & Evaluation)
  Medlock, M. et al.
  http://www.lutin-userlab.fr/gamelab/IMG/doc/Medlock-RITE.doc

  “Using Formative Usability Testing as a Fast UI Design Tool”
  John Schrag. UPA 2006

• User proxies
  http://tinyurl.com/userproxy

• Recurring user studies
  “It’s Not Rocket Surgery - First Fridays in the U.S. Government”
  Nicole Burton, Steve Krug et al. UPA 2010
Discussion: Formative usability testing

• Who is using this?
• How is it working?
• Questions, or problems?
agile qualities

iterative
incremental
continuous
collaborative

During an iteration...

The next few slides show what different groups typically work on during early, mid, and late days within an iteration....
Developer Track (typical)

Next iteration:
• Would you rather:

  Fix bugs, which causes people to perceive the work to be running late

  Work on shiny new features that are perceived to be taking the product to its release goals.
Design Track (typical)

Meetings, bug-tracking
No time for design!

Docs/QA Track (typical)

Desperately try to catch up with the previous iteration
Cultural Change Required

Is the feature DONE?

**TODAY**

Yes, it’s done, the developer just checked in the code. Of course, it still needs to be documented, validated and bug-tested, but it’s all basically there and you know we’ll ship it anyway so let’s just call it done.

**TOMORROW**

Yes, it’s done. The code is in, all the bugs we plan to fix are fixed, the documentation is written, and the product designer validated that it solves the customer’s problem. We can ship it today.

How can you be done after each iteration? Seriously?

You need to change the timing and priorities of activities in each iteration, for everyone.
Developer Track (ideal)

- Code new work in smaller pieces.
- Check in early and often.
- Make sure QA knows what needs to be tested.
- Fix bugs in new code.
- Ensure work meets acceptance criteria with designer.
- Review draft docs
- If time left over, fix legacy bugs or get started on next iteration.

Design Track (ideal)

- Check in with developers to make sure they are building what you expect.
- Groom the backlog.
- Design UI for next iteration.
- Test new code – does it meet acceptance criteria?
- Communicate problems/bugs to developers
- Prepare demos
- Review draft docs
### Docs Track (ideal)

<table>
<thead>
<tr>
<th>Early</th>
<th>Mid</th>
<th>Late</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write rough docs based on design</td>
<td>Gather feedback, finalize docs</td>
<td></td>
</tr>
</tbody>
</table>

- Catch up with previous iteration (polish*)
- Review designs, offer feedback
- Analyze user learning needs, break down required info (concept, task, reference)
- Start writing topics
- Distribute draft topics for review
- Incorporate any changes, corrections
- Prepare and test docs build
- If time left, planning for movies

---

### QA Track (ideal)

<table>
<thead>
<tr>
<th>Early</th>
<th>Mid</th>
<th>Late</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integration/holistic testing</td>
<td>Acceptance</td>
<td></td>
</tr>
</tbody>
</table>

- Keep aware of what your developers are working on.
- Do local-build testing with willing developers.
- Prepare data/tools for upcoming features
- Close any other Resolved defects
- Integration / system testing
- Test Resolved issues as soon as possible.
- Run tests on Resolved Issues to confirm they can be accepted.
- Review draft docs.
- If time left over; test-on-paper imminent stories/tasks.
**Discussion**: Iteration planning

- The next (2-week) iteration has 15 days
- You have designs for 2 stories.
  - Devs have estimated the following:
    - Story 1: Task A (5 days), Task B (1), Task C (1)
    - Story 2: Task D (4), Task E (2), Task F (1)
  - QA estimates a high effort for testing Story 1.
- Your product has significant technical debt.

What should the team commit to?
What’s missing? Communication!

WRONG

Bug tracking software

Dev

Design

QA

Docs
Face-to-Face Communication

- Where possible  
  (Distributed team members discussion below)
- Screen-sharing is good
- Doesn’t have to be a formal meeting

---

Face-to-Face Communication

**Daily Scrum:** Design, Dev, QA, Docs

**Goal:** Prevent surprise. Catch problems fast.
Face-to-Face Communication

Design reviews: Design, Dev, QA, Docs

Goal: Consider other perspectives in design to prevent unanticipated problems later, make estimates possible.

Face-to-Face Communication

Walkabout: Design & Dev, informal

Goal: Make sure dev understands what you meant and is building what you expect.
Face-to-face communication

**Bug Sweeps:** QA & Design

**Goal:** Prioritize defects to prepare for iteration planning.

---

Face-to-face communication

**Site Visit Demo:** Design, reporting to team (after scrum)

**Goal:** Give the team an understanding of users’ problems from an on-site visit
Face-to-Face Communication

Estimate request: Design & Dev

Goal: Entering design-related Tasks. Prepare for the next iteration planning.

Face-to-face communication

Team demos/Retrospectives: whole team

Goal: View past progress. Improve process.
Face-to-face communication

**Iteration planning:** whole team

**Goal:** Everyone knows what they’re working on for next iteration. Check progress against charter. Adjust planning board.
All together – but not in the same place

Richer Communication

• Face-to-face conversation with a physical whiteboard
• High-resolution, large-screen videoconference with a virtual whiteboard
• High-resolution, large-screen videoconference
• Low-resolution, small-screen videoconference
• Phone call using high quality hardware and a land line (=clear connection)
• Phone call using poor quality hardware and VOIP (= noisy connection)
• Instant messaging and real-time text chat
• Asynchronous messaging/voice over with video capture
• Wikis and electronic discussion boards
• Email

Poorer Communication

adapted from Pete Deemer. The Distributed Scrum Primer 1.0

All together – but not in the same place

Richer Communication

• Chartering/Project Kickoff
• Project Retrospective
• Initial Iteration Planning
• Iteration Planning
• Iteration Demo/Retrospective
• Design Review
• Estimate request
• Virtual walkabout
• Bug sweep
• Scrum

Poorer Communication
Not quite face-to-face communication

Each team member should ideally have
• webcam & high-quality headset/mic
• desktop-sharing & -recording software

Each team location should ideally have
• “always-on” videoconference between each team location
• whiteboarding software & tablet
Incremental implementation

Getting to complete workflows, one DONE at a time

agile qualities

iterative
incremental
continuous
collaborative
What if your story is too big?

• The work can’t be completed in one iteration
• You need to break it down
• What are the criteria?
• How do you prioritize the feature cards?

Big Design - Waterfall

• One big design document contains everything
• Everyone signs off
• Dev builds it until they run out of time
• QA doesn’t test until Dev has run out of time
• Result:
  – whatever they built first is completed
  – details are left out, quality issues identified too late
  – holes are left in the design
• Much of your design effort is wasted
Big Design - Agile

• Break the story into small pieces, where each piece confers incremental value *to the user*.
• Determine the minimum first step
• Schedule the pieces in order of importance
• Design incrementally, as if the next piece were the final one
• Change your future plans between iterations if you have learned new things

Benefits of being incremental

• When development runs out of time/resources, the shipped solution
  – Delivers maximum value
  – Has a complete design without holes
  – Has much higher quality
  – Has no wasted design work
Mistakes to avoid

• Designing all the detail up front
• Not *thinking* about the full design up front
• Not breaking things down far enough
• Not delivering a complete (sub) story each iteration – “now the user can...”

Example 1: Doorway

• User story: The user can get in and out of her house easily.
• Completion Criteria:
  - Secure
  - Insulated
  - Lets light in
  - Allows large furniture items to pass
  - Fits with house décor
  - Works even without keys
Example 1: Doorway

• Initial Rough Design:
  - Beautiful Colonial Door
  - Unbreakable translucent window
  - Programmable digital lock
  - Steel deadbolt
  - Metal-clad on the outside
  - High R-Value

Example 1: Doorway

• What is the minimum work that will give the user incremental value towards their goal?
• What needs to be designed for that?

• What is the next smallest item that will give the user an added capability?
• What needs to be designed for that?
Fitting this to your process

• The purpose of incremental implementation is to get feedback early and often.
• After each iteration, gather feedback.
• These questions can affect your breakdown:
  Who evaluates your product?
  Is it always the same people?
  Are your target users internal or external?

Fitting this to your process

You may get feedback from:
• Internal ‘expert users’
• Beta testers under NDA
• The general public (after release or open beta)
• Internal users in a protected ‘sandbox’
• Internal users after general deployment
Fitting this to your process

Before releasing, consider:

• Are you getting the feedback you need?
• Is there enough completed for an external user to evaluate?

Sometimes you may want to hold back certain work until more is done.

Make it easier for the team

• Write *staged specifications* -- a best guess at breaking the design into 1-iteration Story increments
• Then “break” the Stories with developers into Tasks. Remember: they own the Tasks. But you need to know how to map those back to Stories & Capabilities.
Exercise: Stove

- As part of building an entirely new kitchen, your team is delivering a natural-gas kitchen range system.

- What is the absolute minimum first deliverable to internal users/testers?
- What is the absolute minimum first deliverable to external beta users?

Stove exercise. Continued

- Gas burners
- Controls for the burners
- Range hood
- Fan for range hood (requires electricity)
- Light for range hood (requires electricity)
- Automatic sparker to light burners (requires electricity)
- Cook timer (can be set to ring after some length of time)
- Oven (box with door and interior burners)
- Controls for oven (sets temperature)
- Convection fan for oven (improves evenness of cooking)
- Self-clean feature (turns oven on very high for 1 hour)
- Safety lock for oven (prevents someone from opening door when it is too hot)
- Oven light (illuminates interior of oven)
- Burner covers (hides the burners for aesthetic reasons)
- Gas leak alarm (sounds if a gas leak is detected)
- Gas line cutoff (allows you to turn off the gas where it enters the range)
- Electrical connection (required for timers, lights, alarms, and sparkers)
- Drawer for holding pots and pans
- Toothpick holder
- Ability to use bio-gas
Agile UX tactics

Day to Day
Parallel track workflow
BREAK
Staged stories
Design chunking

Timing of communication
Timing of agile UX activities
Incremental implementation
Incremental design

Design chunking
Timebox your design work
agile qualities

iterative
incremental
continuous
collaborative

once you have the high-level big picture...

(Product) Vision
(Release) Mission
...set design goals at the
user workflow level

Capability

Then, design to solve for a few
goals in each iteration
Capability goals

• Description of user’s problems to solve for a workflow or user story + acceptance criteria
• For each iteration, solve a few goals
• Defined through chunked research
• Used to chunk designs
• Used to define “done” of design (& also implementation)
Example: Brush Resize
Brush Resize design goals

- First 5 minutes: learn without documents
- Resizing without Brush Editor
- One control for size, not 2-5
- Keep focus in-canvas
- Fewer dialogues (covering the work)
- Stylus only (no keyboard)

How do you research & usability test?

- Break Capability into smallest Stories
- To design a capability/story over >1 iteration, break a design into chunks
- Which design goals can you investigate with early working versions/prototypes?
- Mix & match design chunks in investigations: mini-research, usability test, & iterate on mini-prototype
Size and detail of design

• Split the design investigations into chunks based on available user resources
• Use user proxies (internal & external) to give you feedback on code/prototypes that doesn’t do full workflows
• Add detail to design for each chunk at latest possible moment

Timing of design activities

• Just as you staged the implementation, you can stage your design activities
• There are criteria that will help you place certain design chunks into early-, mid-, and late-stage chunks
During an project...

The next few slides describes when to work on different design chunks within a project cycle.

**Early-stage design chunks**

To investigate/prototype/test:
- solutions that don’t require domain knowledge
- operation-level, not workflow-level of user task
- prototypes that need a lot of facilitator intervention
- solutions you can check independently of each other (no dependency on other design goals)
- prototypes outside of the main build (no change penalty)
- **Fundamental** designs (other designs will be built on top of these designs)
- Designs that will be re-used in other contexts
Early-stage design chunks

Examples:

• Algorithmic prototypes (SketchBook Pro rotate)
• Specific type of web input field or interaction (type-ahead text entry, date input, credit card input, etc.)
• New types of interaction (iPhone/iPad gesture-based colour picker)

Litmus test: Is this a prototype where you would learn something by driving it?

Brush Resize design goals

• First 5 minutes: learn without documents
• Resizing without Brush Editor
• One control for size, not 2-5
• Keep focus in-canvas
• Fewer dialogues (covering the work)
• Stylus only (no keyboard)
Example: Brush Resize with hotkey

Disposable code prototypes
- Resizing without Brush Editor
- One control for size, not 2-5
- Keep focus in-canvas
- Fewer dialogues (covering the work)

Late-stage design chunks

To investigate/prototype/test:
- solutions that require specific contexts (specific users, environment- or device-dependent)
- solutions that depend on other technology or build on a prior implementation
- workflow-level of user task
- discoverability or learnability goals
- higher fidelity/lower facilitator intervention
- Hub designs (designs that depend on other fundamental designs)
Late-stage design chunks

Examples:

• First-experience or installation investigations
• Designs from multiple agile teams (same product)
• Need to support next Mac OS X (or iOS)
• Hub
  (SketchBook Pro Brush Palette = Brush Resize, Custom Brushes, and Brush Defaults Editor)

Litmus test: Is this a prototype where only a specific user or context can validate?

Example: Brush workflow

Disposable code prototype

• First 5 minutes: learn without documents

Combines 3 Capabilities

• Brush workflow prototype = Brush Resize + Custom Brushes + Brush Palette
Mid-stage design chunks

To investigate/prototype/test:
• solutions that require partial domain knowledge (user proxies are acceptable)
• solutions that combine previous implementations
• part of a workflow known to end-users
• gather workflow-level scenarios for late-stage design
• Parallel designs (designs that can be developed independently)
• other designs that aren’t obviously early- or late-stage
Mid-stage design chunks

Examples:
• Graphic design (during interaction design)
• Emotional response to the look of a UI (e.g., does a look reinforce brand characteristics?)
• Mix and match design chunks per session (SBP: Interactive Brush Outline + 3 other design chunks with animation students)

Litmus test: Is this not an early- or late-stage design chunk?

Example: Brush Resize graphics

Variations created as interaction designed.
• Stylus only (no keyboard)
• Keep focus in-canvas
• Fewer dialogues (covering the work)
Recurring user studies

Set up regular, recurring usability investigations
– Contextual inquiry + usability test whatever you have available at the time
– Can apply to in-house, on-site, or remote

Timeboxed design encourages:
– continuous end-user input
– lower-fidelity prototypes
– faster movement through design solution space

Example: Recurring user studies

– In-house. End-users brought in every 2 weeks
– On-site. Drove to Detroit every month.
– Remote usability testing. Every 3 iterations.
Combine user investigations

Contextual inquiry & usability testing in same session
– The most expensive cost of user investigations is scheduling
– Get more data in less time
– Test and investigate much smaller chunks
  (15 minutes of work)

“Formative usability investigations for open-ended tasks”
Desirée Sy. UPA 2006

Formative design test tips

• Schedule the sessions with enough time between each to iterate
• Know the design goals prior to the sessions
• Start from the simplest solution to the problem
• Prototype the solution in the fastest, most malleable way to validate the solution
• If high-fidelity (or in code), prototype changes during the test run are not production code
When to change the prototype

• At the end of each session, go back to the design goals -- are you done yet?
• Can you change the prototype to get closer to done in a way where you can validate effect?
• If a problem blocks investigating other design goals, make the change
• If you uncover a solution that doesn’t require validation, don’t change the prototype

Engaging the team

• Developers working on a story should see at least one usability test for that story
• Interaction designers should be observing all tests of their designs
• All team members should be invited to observe
• Key test results should be “re-enacted” for all team members
Implementation order ≠ Research

• The order that you researched the design chunks should not influence the order of implementation

• Use the principles discussed during Staged Stories to determine incremental value.
Further references

- **User proxies**
  http://tinyurl.com/userproxy

- **Combine user investigations**
  "Formative usability investigations for open-ended tasks"
  Desirée Sy. UPA 2006

- **Recurring user studies**
  "It's Not Rocket Surgery - First Fridays in the U.S. Government"
  Nicole Burton, Steve Krug et al. UPA 2010

Exercise: Design chunking
Inventory system used on factory floor of warehouses.
- **Capability:** Line manager on factory floor can scan an item with a UPC barcode into database.
- **Scenario:** Line managers take samples of a manufactured part at different points on assembly line, and put the samples in a box with a UPC code. (Before: manual entry)
Design goals for Scan Sample

• No errors in the inventory database
• Reduce errors caused by manual data entry
• Even if the scanner breaks, it must be possible to add the part and time info into the database
• Line manager needs to be able to move to different stations in a warehouse
• Line manager has to be able to do this without asking for/looking up help
Exercise: Design chunking
Which design goals could be researched or tested:

- At your local office, with QA & support?
- At the warehouse, with a line manager?
- In short sessions by line managers at the 2-day training course, between breaks?
- Remotely with line managers?
“To make design activities incremental & continuous, craft as many early-stage design chunks as possible.”

Wrap up
Summary and Course retrospective
agile qualities
iterative
incremental
continuous
collaborative

Tutorial leaders

John Schrag
john.schrag@autodesk.com
@JVSchrag

Desirée Sy
desiree.sy@autodesk.com
@DesireeSy