Managing UCD Within Agile Projects

Mark Detweiler | SAP | mark.detweiler@sap.com

THE GOALS OF THIS ARTICLE are to share some insights from managing user experience (UX) professionals involved in software-development projects that follow an Agile development methodology, and to stimulate thoughts about ways to anticipate and address potential challenges that Agile approaches might pose for a UX director or manager.

Over the past decade, traditional, plan-driven software development methods have been supplemented with a growing number of newer, more nimble methods, including: Extreme Programming [1], Agile/Scrum [2, 4, 9], Crystal Methods [4], and Adaptive Software Development [5]. While these methods vary in terms of their specific approaches and unique contributions, they all attempt to cope with accelerated rates of change in highly competitive software markets.

These approaches break up large development projects and deliverables over many months into smaller chunks of work over much shorter durations, e.g., often into cycles of two- to five-week "sprints" [9]. Key benefits associated with these approaches are well documented in the literature [1,2] but focus mostly on creating higher-quality code quickly with smaller teams. For the purposes of this article, I will focus on Agile initiatives, but I openly acknowledge that specific details may vary considerably even among these newer approaches. Before addressing Agile challenges, it is necessary to consider what I mean by user-centered design (UCD), even though most readers will be quite familiar with one or more approaches.

Most flavors of UCD being practiced tend to regard UCD as both a general philosophy of focusing on people or "users" throughout the product life cycle and a set of methods and tools used to produce products with this focus. SAP has borrowed from the rich tradition of UCD [6, 7, 8] and promotes four fundamental principles:

1. Focus on "real" end users and engage them early and continuously throughout the product life cycle.
2. Validate UI requirements and designs by observing, measuring and recording end users.
3. Design, prototype and develop UIs iteratively.
4. Understand and design for "holistic" user experience.

Figure 1 depicts three general iterative phases of UCD. Phase 1, Understanding Users, involves observing and interviewing end users and other stakeholders. It is needed to gather requirements, e.g., with field research, focus groups, and interviews. The deliverables in this phase are detailed user profiles and rich descriptions of end users' workflows and pain points. Phase 2, Define Interaction, takes the outputs from Phase 1 and creates use cases [3] from the user research. These use cases are used as input to Phase 3, Design UI, where prototypes are iteratively created and evaluated.

A key difference between traditional plan-based and Agile development is that Agile projects do not spread the UCD phases over the entire development life cycle. Instead, they often repeat all three phases across sets of sprints or milestones. With the foregoing framework in place, I will next point out a few challenges that apply across all three UCD phases, with special emphasis on Agile projects, and then present some tips for addressing them, followed by phase-specific challenges and tips.

Challenges Applicable Across All UCD Phases

- Agile projects operate under highly compressed time scales. They tend to lack traditional project-management processes and skills, relying heavily on team self-governance. This often means that UX managers must become more actively involved than they normally might to insure that UX activities are regularly included in team-based planning and scheduling. Communication and coordination across teams may also fall more heavily on UX members responsible for insuring compliance to UI style guides and accessibility legislation. The compressed time scales may also make it difficult to get access to the right customers at the right times—to gather requirements and feedback on prototypes and working builds. Contrary to strong textbook recommendations to have close customer partners and regular feedback, many Agile projects engage customers only sporadically.
- Agile projects may require more UX headcount to do the same work, since many tasks that are performed sequentially in plan-based projects must be done in parallel.

General Tips

- Leverage support from key allies and influencers, particularly development executives, and tailor UCD activities to match each culture's unique needs.

- Proactively calculate UCD efforts and costs before projects begin. This will increase the likelihood that UCD activities will be adopted and supported.
- Lobby vigorously for sufficient headcount and resources; demonstrate UCD value with small wins to justify larger investments, for example, in staff, lab resources, and travel budget.
- Network with other UX managers within and outside your organization who have been successful in Agile projects to learn from their success stories.
- Help your staff keep project teams well informed of project status, for instance, when prototypes are posted, modifications are made, feedback is requested, and usability studies are running.
- Make all "work-in-progress" and deliverables highly visible to project teams to promote sustained dialog; have development teams routinely look at work in progress. Post workflows, wire-frames, screenshots, etc., with notes to ask questions, clarify, challenge, and propose alternatives.
- Share success stories with other development teams, and create your own UCD "reference customers" willing to help promote UCD with good examples.
- Encourage UX staff to volunteer to facilitate brainstorming and review sessions; the person facilitating at the board usually sets the tone and can help drive the agenda/process.

Phase 1 Challenges

- In contrast to plan-based approaches, there are often few if any background documents or specifications to help provide context for requirements, e.g., market requirements documents (MRDs) or product requirements documents (PRDs).
- Often there is little support for or too little time allocated to conduct genuine field studies (observing, interviewing, etc.) or requirements-gathering to discover real end-user work practices, workflows, pain points, etc.

Phase 1 Tips

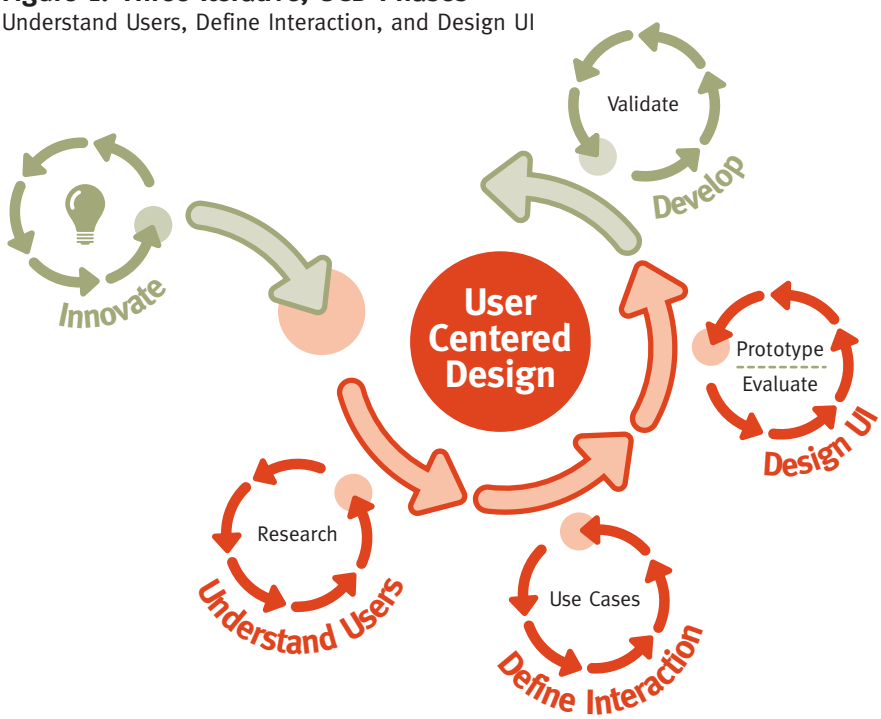
- Lobby hard to have dedicated sprints/milestones allocated for gathering requirements, especially at the beginning of an Agile project. Involve developers in synthesizing the data gathered from customer visits.

  Where possible, try to anticipate activities that can be done before a project is officially kicked off—reading background literature, gathering information on competing products (where appropriate), conducting early customer interviews, site visits, focus groups, etc.

Phase 2 Challenges

- Most Agile approaches tend to eschew as much documentation as possible. As

Figure 1: Three Iterative, UCD Phases
Understand Users, Define Interaction, and Design UI



such, they often do not see value in writing down use cases.

Phase 2 Tips

- Promote Agile-friendly use-case methods [3] so that time and effort are not wasted. Demonstrate that by explicitly writing down users' goals, the steps needed to achieve them, and the data needed to support the steps it is possible to accelerate the production of prototypes and working code.

Phase 3 Challenges

- UI consistency may be undermined as independently empowered teams evolve code in parallel, without coordinating with each other.
- Not enough time or staff may be available to code new UI platform components/widgets in time for a particular release. Further, new UI components/widgets that need to be shared efficiently across teams may require special UX coordination to ensure that separate teams communicate effectively with each other.
- Not enough time is budgeted for evaluating and testing prototypes and working builds with representative end users.
- Code generated during sprints/milestones is often too unstable to conduct usability tests on, even if it was promised and scheduled.
- Not enough time/effort is allocated to respond to insights gathered from user feedback and usability evaluations.
- UI designers may start to identify very closely with the goals and needs of their respective project teams and lose their "objectivity" and user focus.
- Not enough time is available to prototype alternatives or to iterate designs.

Phase 3 Tips

- Show prototypes to design partners and target end users to gather their feedback early and often.
- Encourage feature teams to watch user studies—to foster "buy-in" and to leverage group knowledge; hold mandatory review sessions after every feedback session/usability test.
- Develop tools to help prioritize UCD work—use an Importance x Difficulty matrix to drive usability fixes, e.g., importance (high, medium, low) crossed with difficulty to fix (high, medium, low).
- Closely monitor builds and flag gaps between UI designs and working code early; push hard to have gaps closed quickly.
- Combat time pressures and difficulties of accessing appropriate end users by running remote usability tests.
- Conduct design reviews with other UI designers to assess designs, to share best practices, and to fight tendencies of having designers identify too closely with goals of their respective feature teams.

ABOUT THE AUTHOR

Mark Detweiler is vice president of SAP UX Methods at SAP Labs in Palo Alto, California. He and his team are responsible for defining and supporting SAP's UCD process worldwide. Prior to joining SAP, Mark spent 25 years contributing to and managing HCI research and development at companies including Adobe, Ariba, Interval Research, Oracle, Bellcore and Honeywell.

REFERENCES **1.** Beck, K. (1999). *Extreme Programming Explained.* Reading, MA. Addison-Wesley. **2.** Boehm, B., & Turner, R. (2003). *Balancing Agility and Discipline.* Reading, MA: Addison-Wesley. **3.** Cockburn, A. (2000). *Writing Effective Use Cases.* Reading, MA: Addison-Wesley. **4.** Cockburn, A. (2006). *Agile Software Development: The Cooperative Game (2nd Ed.).* Reading, MA: Addison-Wesley. **5.** Highsmith, J.A. III (1999). *Adaptive Software Development.* Dorset House Publishing Company. **6.** Norman, D. (2002, reissue). *The Design of Everyday Things.* New York: Basic Books. **7.** Norman, D. (2005). Human-Centered Design Considered Harmful. *Interactions*, (July-August), pp. 14-19. **8.** Norman, D., & Draper, S. (Eds.) (1986). *User Centered System Design: New Perspectives on Human-Computer Interaction.* Hillsdale, NJ: Lawrence Erlbaum Associates. **9.** Schwaber, K. (2004). *Agile Project Management with Scrum.* Redmond, WA: Microsoft Press.