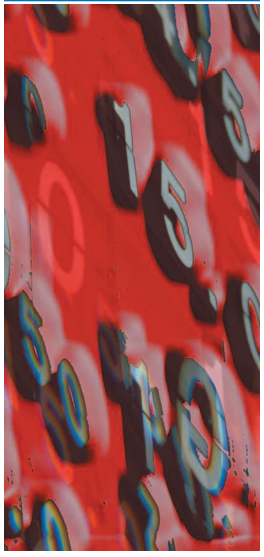

UCD in Agile Projects: Dream Team or Odd Couple?

Paul McInerney > IBM Toronto Lab > paulmci@ca.ibm.com
Frank Maurer > University of Calgary > maurer@cpsc.ucalgary.ca

IMAGINE INTERVIEWING for the position of UCD specialist in a company that uses an agile software development process. Could you answer questions such as, "how does UCD fit in an agile process?"

UCD specialists may be horrified when they first hear about agile methods [1]. Agile literature emphasizes developers delivering production code quickly rather performing extensive analysis and design (e.g., UCD) prior to coding. However, the agile literature indicates that this perception is simplistic and misguided. Examining professional practice, as we do in this article, paints a different picture of how UCD and agile practices coexist in a development team.

INTRODUCTION TO AGILE METHODS. The agile approach is quickly becoming mainstream in the software industry. The agile community is defined by a core set of beliefs and practices. The most well-known articulation of agile beliefs is the Manifesto for Agile Software Development (www.agilemanifesto.org), which states:

"...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

..."

The most widely used agile approaches are Extreme Programming [2] and Scrum [5]. Information about others (including Adaptive Software Development, Agile Software Development, DSDM, Lean Software Development, Feature-Driven Development, Agile Modeling) can readily be found on the Internet.

The following paragraph highlights the interactions occurring in an idealized agile team by providing a glimpse into a typical flow of work. Italicized terms are key agile terms.

At the beginning of an *iteration* (or *Scrum Sprint*), members of the team sit down with their *on-site customer representative(s)* to discuss what features they'd like included in their application. While a bit ambiguous in the agile lingo, customers include anyone who has a stake in the development project (including clients and users). Customer requests are captured as *user stories* on index cards. A user story is a description of a feature of the software system that has business value in the eyes of the customer representatives. User stories play a similar role to use cases, scenarios, or requirements lists that are used in non-agile methodologies. As the information captured on an index card can't be a complete feature description, user stories are treated as a reminder for further communication with the customer representatives. Based on effort estimates for stories provided by the developers, the customers select the highest-priority stories for the upcoming iteration. Others are placed in the *product backlog*. During the iteration, the programming team divides the work among *pair programmers* and completes the entire *test-driven development life cycle* for each story: Automated acceptance tests drive the development of unit tests that, in turn,

drive the development of production code that has to make all tests pass. Regression testing supports *refactoring* of the software design. To enable easier face-to-face communication, the entire team works at computers on a boardroom-type table. There is a 15-minute *daily scrum* where each member provides a brief update on her progress, her plans and any obstacles blocking her work. At the end of the iteration, the team is supposed to deliver *potentially shippable product functionality* for an acceptance review. The iterations continue until the customer determines that the incremental costs for additional features cannot be justified by the increased business value.

In the following, we'll discuss how UCD specialists found their role in agile environments.

INTRODUCTION TO THE CASE STUDY. The authors interviewed three UCD specialists who were working on their first agile project. They all have degrees with a UCD/HCI specialization and prior UCD experience.

- TB works on medical instrumentation products using the Industrial XP methodology.
- MG's project involves replacing a character-based UI to a supply-chain-planning product. The company uses an agile approach developed in-house.
- PV is an independent consultant whose client is a start-up company developing tools to support agile development. The company uses an agile approach developed in-house.

The remainder of this article compares agile literature to the case studies under following headings: making the case for UCD, understanding users, UI design, and evaluating design usability.

MAKING THE CASE FOR UCD. Like other development methods, the agile literature does not identify a distinct UCD role, so the onus remains on UCD to justify and define its role on the team. Table 1 shows common UCD claims and how they might be countered by someone familiar with agile literature.

Table 1: Common UCD claims and how they might be countered by someone familiar with agile literature

UCD Value Proposition	Agile Response
UCD represents users within the development team.	Agile teams include customer representatives.
UCD provides *specialized* skills in *UI design*.	Agile approaches prefer generalists and discourage extensive upfront design work.

While the preceding table paints a depressing picture, our case studies told a different story.

- TB: His company's centralized UCD group supported a proposal to pilot agile methods and dedicated TB to the pilot team. TB's UCD role is part of the product management team, which includes product managers,

domain experts, a technical writer, and two testers. As part of the "everyone pitch in" ethos of agile projects, TB helps write acceptance tests, and other product management team members help run usability tests.

- MG: MG's company has a centralized UCD group with an executive leader. That leader mandated that UCD staff be routinely included on projects. Receptivity to this directive varies. MG supports two teams; the team discussed in this article was receptive. Compared to non-agile projects, timesharing is more difficult. MG has not experienced much blurring of her UCD role.
- PV: PV is an independent consultant whose client is a start-up company. The company founder sought UCD skills because of an exposure to UCD at a prior company. PV works mostly remotely and uses instant messaging. In addition to his traditional UCD duties, he helps solve technical Web development issues.

UNDERSTANDING USERS. This section examines understanding users as an input to subsequent design activities. The agile literature perspective can be characterized by the following points:

- Favors developers working directly with customer representative(s) to understand requirements.
- Based on developer recommendations, customer representatives make final decisions on how the system should behave.
- Complete only a preliminary analysis before beginning coding and clarify customer needs as questions arise during the coding.

Our practitioners encountered practices described above but also experienced significant deviations. They reported that the "customer representative" was often an employee of the software vendor. This person could be a domain expert, a former employee of the customer, or someone responsible for product direction. Our practitioners often found they were able to extend agile practices with traditional UCD approaches, including personas. As well, UCD was able to undertake significant "up-front analysis and UI design." Here are some specific experiences:

- TB: By coincidence, UCD had completed contextual enquiry and personas before the agile project was launched. TB recommends this up-front analysis prior to project start-up.
- MG: Inclusion of UCD in writing customer stories is variable.

UI DESIGN. The agile literature suggests UCD UI design practices need adjustment, as shown in Table 2.

Our case study participants confirmed that their UI design practices required adjustments, especially to adapt to iterative development. They gave the following examples of the scope of UI design addressed in a single story:

Agile Method Attribute	Possible Implications for UI Design
Each iteration focuses on vertical slices through the application.	UI design focuses on a small piece of the application that progresses rapidly from concept to code.
There is little notion of a "UI designer" specialty.	UI design may be more of a "team effort.".
Pairs of programmers work out software design on the spot as they program.	The UI designer needs to be "on call" to participate in ad hoc discussions.

Table 2: Agile literature suggests UCD UI design practices need adjustment.

- A single dialog box or area of the screen, e.g., an interactive graph.
- A *Find* dialog box, including determining the places where this dialog should be launched from.

Here are some specific experiences:

- TB: He spends most of his time on the current iteration, especially answering questions about his UI design and discussing or reviewing programming work in progress. He spends the rest of the time working on prototypes for stories for the next one or two iterations. He also maintains an unofficial visionary prototype as a roadmap beyond the current iteration. TB reports as an example of the "all pitch in" ethos that the project manager has taken on the role of reviewing design details for UI guideline compliance.
- MG: She tries to generate multiple design options and engage programmers in discussions about which options best fit the story. The approach contrasts with that of developers who tend to create hi-fi prototypes that only explore only one option. MG highlights that the lack of UI design ownership means that everyone wants to be involved in design, which can lead to design by committee. As well, when there is a nontrivial UI design change, she needs to convince everyone.
- PV: PV reported misgivings about design decisions being made in early iterations without a view of the entire UI. Over the course of many iterations, he sees shortfalls in prior work but doesn't have the documentation of the rationale for the earlier design. In the future, he'll consider writing design rationale or design patterns. PV also observed that as the project progresses, he can deliver more design per iteration because the team can reuse design and code from prior iterations.

EVALUATING DESIGN USABILITY. This final section examines evaluating design usability. Some differences between UCD and agile approaches are shown in the Table 3.

Our practitioners were sometimes able to extend agile practices with traditional UCD approaches. Limited success seemed due to traditional barriers, namely schedule impact and difficulties making design changes late in the cycle.

ACKNOWLEDGEMENTS AND DISCLAIMER

Thanks to Mahitha Gokulachandra, Tom Bellman, Katherine Marshak, Pawan Vora, and Jeff Patton for their time for the interviews and their feedback on this article. The views expressed in this article are those of the authors rather than their employers.

ABOUT THE AUTHORS *Paul McInerney is a senior member of the UCD team for the DB2 database product at IBM. He has written or presented on topics such as writing UI specifications, managing UI design, scenarios, tracking usability defects, and integrating usability engineering into traditional software development methodologies (published in the anthology* Design by People for People: Essays on Usability).

Dr. Frank Maurer is the head of the e-Business Engineering group at the University of Calgary. The work of his group focuses on agile methods and Web engineering. A recent line of research deals with combining agile methods with ideas from human-computer interaction and user-centered design. More information can be found at http://ebe.cpsc.ucalgary.ca/ebe. Dr Maurer is the co-founder of the Calgary Agile Methods Users Group (http://www.agilenetwork.ca/camug) and the Canadian Agile Network (http://www.agilenetwork.ca).

UCD Approach	Agile Approach
Use about five users that fit predefined user profiles	Put system increments into production ASAP to get real-world feedback
Start evaluating using low-fi designs	Evaluate production-ready code at the end of each iteration
Use methods such as hands-on testing and collect metrics, such as efficiency and satisfaction	Demonstrate the working code to obtain an accept or fix verdict

Table 3: Some differences between UCD and agile approaches

Here are some specific experiences:

- TB: TB's team conducts empirical testing with real users at least once for every internal release (three months). Little usability testing is done per iteration. They use paired observers for tests, adapting the pair programming practice. Originally, they had trouble getting design changes made, but recently, the team decided to devote two weeks from every internal release to making changes resulting from usability testing. Sometimes they conduct informal usability testing in the programming area with the latest code, which engages the interest of programmers working nearby.
- MG: Her testing is similar to non-agile projects. However, testing is focused on feeding changes into the next project plan (as opposed to fixing this release).
- PV: Usability testing is informally done every few iterations. Testing might involve showing customers or customer proxies prototypes. The pace and timeframe of iterations makes testing more difficult than in non-agile projects. He is still working on convincing the project sponsor to spend time on more testing.

CONCLUSIONS. Agile methods have a distinctive development culture that, at first glance, seems not to be hospitable to UCD. However, all our UCD practitioner reports were positive. Certainly, work-life aspects were positive-they felt actively engaged in a common goal. Our participants said the resulting UI designs may or may not be better but are certainly no worse. These case studies can be seen as interim reports: Agile methods are still maturing, and no case study participant had yet shipped their first agile project. To keep informed about recent discussions, visit the Yahoo discussion group called "Agile Usability" [4].

REFERENCES **1.** John Armitage: Are Agile Methods Good for Design?, ACM Interactions, p. 14-23, January/February 2004. **2.** Kent Beck, Cynthia Andres: Extreme Programming Explained: Embrace Change (2nd Edition), Addison-Wesley, 2004. **3.** Alan Cooper: The Inmates Are Running the Asylum: Why High Tech Products Drive Us Crazy and How to Restore the Sanity, SAMS, 1999. **4.** Jeff Patton (owner/moderator): Agile Usability Discussion Group, http://groups.yahoo.com/group/agile-usability (last visited: Dec 2004) **5.** Ken Schwaber: Agile Project Management with Scrum, Microsoft Press, 2004.