

## **Designing with an Agile Attitude**

*Rebecca J. Wirfs-Brock*

Vol. 26, No. 2  
March/April 2009

This material is presented to ensure timely dissemination of scholarly and technical work. Copyright and all rights therein are retained by authors or by other copyright holders. All persons copying this information are expected to adhere to the terms and constraints invoked by each author's copyright. In most cases, these works may not be reposted without the explicit permission of the copyright holder.

IEEE  **computer society**

## Designing with an Agile Attitude

**Rebecca J. Wirfs-Brock**

*Attitude is a little thing that makes a big difference.* —Winston Churchill

**G**ood software designers share many traits, habits, practices, and values, whether they work on agile teams or not. Many designers value design simplicity, communication, teamwork, and responsiveness to stakeholder needs. So what distinguishes agile design from other design processes? Do successful designers working on agile projects need radically different techniques and skills?



### Agile Design Supports Existing Values

As a reviewer of experience reports at several agile conferences for the past five years, I've vicariously experienced the joys, struggles, and triumphs of several successful design teams. Most didn't ignore their corporate culture or existing design context even though they shifted priorities and added many new practices. Adopting agile development seems to go hand in hand with articulating what you value, then finding ways to improve on what you already do well.

Jon Spence, a principal engineer at a medical-instrumentation company, sums up his initial attraction to agile development ("There Has to Be a Better Way," *Proc. 2005 Agile Conf.*, IEEE Press, 2005, pp. 272–278):

*There's nothing wrong with plan-driven, waterfall-based, document-centric approaches.*

*They're just not suited to controlling complex activities like software development. We needed to adopt agile software development because it's the best technique for the activities we're trying to manage and control.*

After studying literature, talking to agile thought leaders, and thinking about how agile practices might fit in his company's development process, Jon and a handful of colleagues launched an agile initiative. They proposed adopting new practices that seemed radically different from their current ones:

- forming small development teams;
- giving each team freedom to adapt its development process;
- emphasizing simple design—designing only for current, not speculative, needs;
- developing in short, two-week iterations; and
- reflecting at the end of each iteration.

They followed test-first development on all new and changed, nontrivial, non-GUI code. They refactored code to maintain simple design only when that code had unit tests in place.

Initial reactions to their proposal were largely supportive. Yet some wondered about how pair programming might impact productivity and whether it would disrupt the status quo of solo development. Management viewed refactoring as having the potential for gold plating. So, they proceeded with the understanding that these practices would get a fair trial and were subject to change as they learned and

applied them. In addition to having project management support, Jon asserts, “it was equally important to have friends and allies in management, both immediate management and upper management.”

Jon’s team came from a culture that values measuring productivity and making process improvements. So, as part of adopting agile practices, they closely monitored the amount of rework, the speed with which they completed tasks, and how effectively they worked through their feature backlog. As a result, they decided to reduce teams from 12 members to four to six members, add a team of system engineers to support the product manager in defining product features and functionality, and adjust their planning meetings to improve intrateam communication.

### Agile Design Improves through Reflection

Frequent checkpoints let designers learn and refine their software and how they work. Marilyn Lamoreux, a colleague of Jon’s, has recounted how she introduced end-of-iteration reflection meetings into their project (“Improving Agile Team Learning by Improving Team Reflections,” *Proc. 2005 Agile Conf.*, IEEE Press, 2005, pp. 139–144). Initially, some developers believed that meetings weren’t “real work” no matter what the meetings’ agendas were. A common belief was that effective meetings should have a detailed agenda and result in actions, decisions, and to-do lists. Her first attempts at conducting reflections were only moderately successful.

After investigating, Marilyn decided to introduce Conversation Café (<http://tinyurl.com/4kxmcv>), a technique that combines simple ground rules and a talking token to guide conversation. The first time she tried it, Marilyn was stunned by how successful it was. She was able to instigate a thought-provoking design discussion about why some code had errors and how to improve it. Marilyn speculates that this technique worked because it “taught our teams some of the conversational practices that open minds and hearts to new ideas.”

Even following this technique, teams still occasionally have unproductive reflection meetings. Marilyn says, “Learning to reflect is a process that takes persistence, practice, feedback, and adaptation.” And, I might add, determined efforts to seek out

and experiment with ways to make a particular practice effective.

### Agile Happens One Step at a Time

Agile design and development practices don’t have to happen all at once to be successful. David Kane reports on how he incrementally introduced agile development techniques into a team that designs software used by US National Cancer Institute scientists and researchers (“Introducing Agile Development into Bioinformatics: An Experience Report,” *Proc. 2003 Agile Development Conf.*, IEEE Press, 2003, pp. 132–139). David’s team already worked collaboratively on development and had ready access to experimental biologists in a lab across the hall. The team wanted to adopt agile methods, but they also needed to continue to make releases while they learned new techniques. They had varying degrees of knowledge about agile methods and needed time to develop their skills.

One of the first things David did was get their code base under better configuration management. Shortly afterward, he introduced the team to automated testing. As their test case coverage grew, they started refactoring portions of ugly code, cleaning up the design and removing unnecessary complexity and coupling. It wasn’t until more than a year and a half into agile adoption that they introduced code reviews and one-day-a-week pair programming. They wanted code reviews to complement their existing practices and help share knowledge among team members. Each iteration, one

developer would pick two to six classes to review from those he or she created or modified. Others would review the code well before the end of the iteration, giving enough time for revision. They found reviewing and discussing code let them raise design and implementation issues that occurred across larger portions of their code.

### Agility Comes with an Attitude

Developing complex software can be difficult no matter how good designers get at architecture, tooling, or technology. Although agile techniques and practices vary, successful agile designers I know are passionate about producing high-quality incremental design solutions. They aim to design and implement solutions for the current problems at hand simply and efficiently. They adopt practices, tooling, and technology that enable them to produce results. They prefer to connect their design work to real, not presumed, needs and aren’t satisfied with being only head-down problem solvers or coders. They expect to give and take criticism and ask clarifying questions of teammates and other project stakeholders. And they take care not to ignore details that could derail design quality or put their project at risk.

**S**o what does it take to be an effective designer in an agile development environment? Although I don’t find agile development to require drastically different design or technical skills, it does demand teamwork and cooperation. Agile designers need to sharpen their communication and collaboration skills as well as their technical practices. They should value collaboration and collective understanding as much as good design and development practices. It’s a matter of attitude more than any specific technique or process. 

**Developing complex software can be difficult no matter how good designers get at architecture, tooling, or technology.**

**Rebecca J. Wirfs-Brock** is president of Wirfs-Brock Associates. Contact her at [rebecca@wirfs-brock.com](mailto:rebecca@wirfs-brock.com); [www.wirfs-brock.com](http://www.wirfs-brock.com).