TECHNOLOGY

The Winter Getaway That Turned the Software World Upside Down

How a group of programming rebels started a global movement

CAROLINE MIMBS NYCE **DECEMBER 8, 2017**

BER1A / IVECTOR / SHUTTERSTOCK / ZAK BICKEL / THE ATLANTIC

Snowbird, Utah, is an unlikely place to mount a software revolution. Around 25 miles outside Salt Lake City, Snowbird is certainly no Silicon Valley; it is not known for sunny and temperate climes, for tech-innovation hubs, or for a surplus of ever eager entrepreneurs. But it was here, nestled in the white-capped mountains at a ski resort, that a group of software rebels

gathered in 2001 to frame and sign one of the most important documents in its industry's history, a sort of Declaration of Independence for the coding set. This small, three-day retreat would help shape the way that much of software is imagined, created, and delivered—and, just maybe, how the world works.

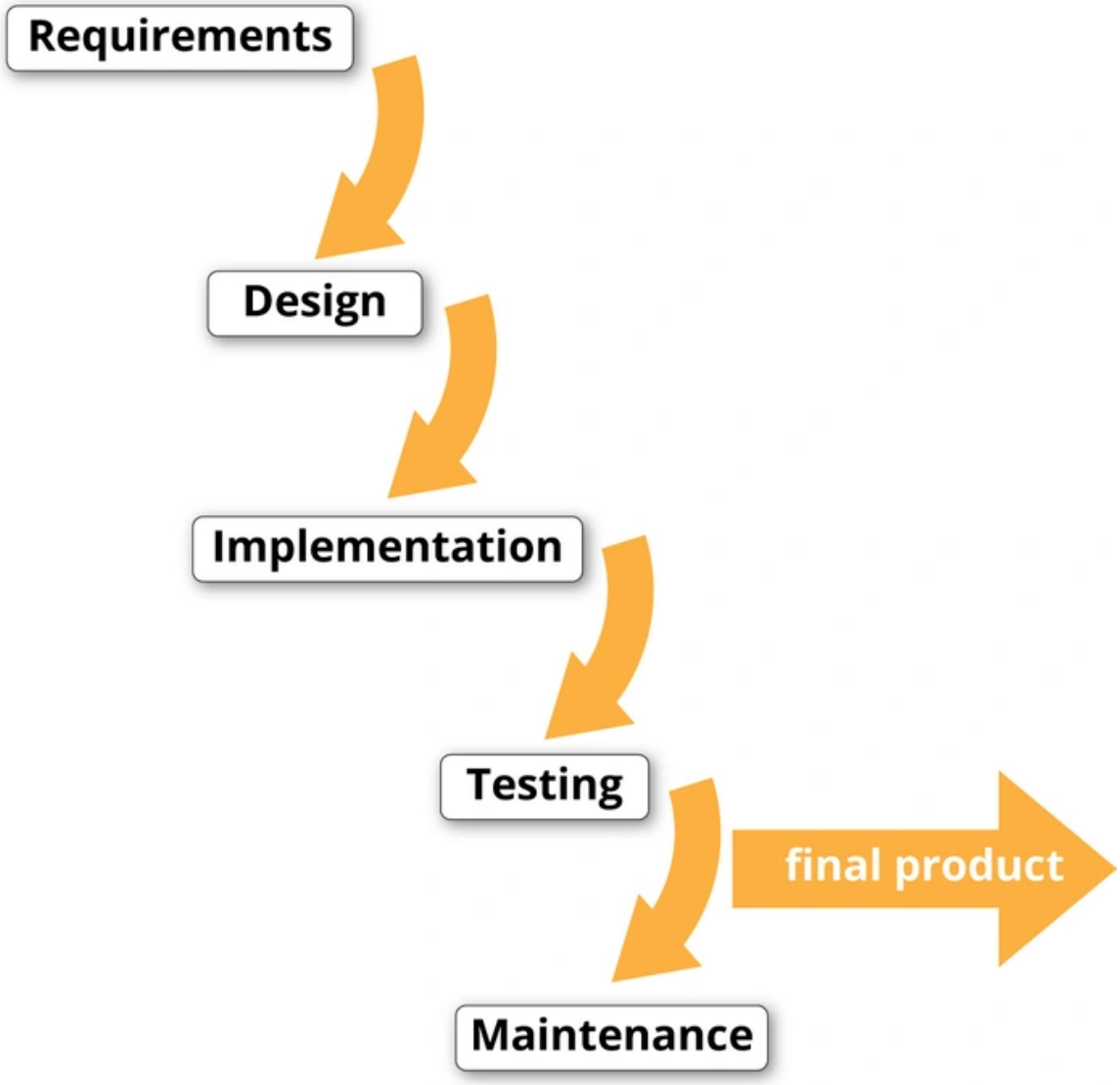
Whether or not you recognize its name, you've probably encountered Agile, or at least companies that use it. Representatives from Spotify and eBay confirmed that both companies currently use Agile, and there's a job listing on Twitter's website for an "Agile Coach." Bread-crumb trails across the internet suggest that many other big-name technology companies have at least experimented with it in the past. And it's not just Silicon Valley: Walmart reportedly began experimenting with Agile years ago. The Agile Alliance, a nonprofit that promotes the use of Agile, counts all sorts of corporate giants—including Lockheed Martin, ExxonMobil, and Verizon—among its corporate members.

Agile's acolytes seem to be everywhere, bringing with them a whole nerd lexicon of tools and tricks to make workplaces more efficient: Think daily stand-ups and sprints. Taken at face value, it may seem like another meaningless corporate buzzword used by project-management types. But it's actually a very specific philosophy, one that is outlined in the four-bullet, 68-word document signed at Snowbird.

* * *

Before software could eat the world, it needed to pull itself out of the deluge. Silicon Valley may be one of the only places in the world where the word "Waterfall" has a slightly negative connotation. In programming, Waterfall is used to describe a way of building software—think a slow, trickling, stage-by-stage process. Under Waterfall, the software project is rigorously designed up front, in the way that one might manufacture a wristwatch.

It worked something like this: Someone would dream up a piece of software they'd like built. Before so much as a line of code is written, the creators write out what they want built and how in a series of long, detailed plans. They craft what's called a requirements document, where they outline everything they want the software to do. Projects then flow downstream, from stage to stage, team to team, until they reach completion. At the very end, the entire new piece of software is tested, given back to the customer, and sent out the door.



A Waterfall process (Courtesy of the Computer History Museum)

Many attribute the origin of this model to a 1970 paper by Winston W. Royce—but there's a big catch: Though a Waterfall-like diagram appears on the second page, Royce's paper doesn't actually endorse building software that way.

A linear approach might work when you know exactly what you want to build, but it can be too restrictive for some projects—software development, as Michael A. Cusumano, the *Sloan Management Review* distinguished professor of management at MIT, puts it, is "really an invention process." "Software engineers or programmers like to go back and forth across those different steps," says Cusumano. "They're not really sequential."

And there's a problem with waiting until the end of a project to test the whole thing, Cusumano points out: If you catch a bug at the last stage, it can be messy—or even fatal—to try to go back and fix it. Some software projects would get stuck and simply never ship.

"People would come up with detailed lists of what tasks should be done, in what order, who should do them, [and] what the deliverables should be," remembers Martin Fowler, the chief scientist at ThoughtWorks, who attended the Snowbird meet-up. "The variation between one software and another project is so large that you can't really plot things out in advance like that." In some cases, the documentation itself grew to be unwieldy. A few of the people I spoke with shared horror stories: an entire bookshelf's worth of requirements in binders or an 800-page document that had been translated across three different languages.

Another Snowbird participant, Ken Schwaber—the cofounder of Scrum and founder of Scrum.org—says Waterfall "literally ruined our profession." "It made it so people were viewed as resources rather than valuable participants." With so much planning done upfront, employees became a mere cog in the wheel.

As the pure sequential model faltered in the 1980s and early 1990s, some companies began experimenting with different ways to work through projects, creating processes that, as the former academic in the field of science and technology studies Stuart Shapiro says, allowed developers to "climb back up the waterfall."

In a 1997 paper on Microsoft, Cusumano and his coauthor Richard W. Selby describe how Waterfall "has gradually lost favor ... because companies usually build better products if they can change specifications and designs, get feedback from customers, and continually test components as the products are evolving."

Around the turn of the century, a few rogues in the software industry began really pushing back. They wanted to create processes that would give them more flexibility and actually allow them to ship software on time. Some of these processes, like Scrum and Extreme Programming (XP), were called "light" or "lightweight" processes. But no one subset had really caught on. So, in 2001, the lightweight guys decided to join forces.

"I think at that point, we were all sort of seeking legitimacy, that we'd sort of been all out on our own doing similar things, but it hadn't really taken off big-time in the community," remembers Jim Highsmith, an executive consultant at ThoughtWorks.

It's unclear who came up with the idea for the meeting that would eventually take place at Snowbird. Many of the participants were leaders in the software community, and a few remember the idea being tossed around at industry meet-ups. When the invitation to the retreat finally arrived, it came via an email, from Bob "Uncle Bob" Martin. Martin, an industry veteran and the founder of Uncle Bob Consulting, runs *The Clean Code Blog* and has a perfect sense of nerd humor: A YouTube video embedded on his website features Martin, among other things, blasting off on an asteroid. Martin says he and Fowler met at a coffee shop in Chicago, where they

composed and sent the email. Fowler doesn't have any memory of their meeting, but says it's "likely" it went down that way.

After running through a few options—like Chicago ("cold and nothing fun to do," wrote Highsmith in 2001) and Anguilla ("warm and fun, but time-consuming to get to")—the group settled on Utah ("cold, but fun things to do"), booking an excursion to Snowbird. There, beginning February 11, 2001, the men—and they were all men—would hit the slopes and talk software processes.

* * *

James Grenning, the founder of Wingman Software, remembers a blizzard. "We were snowed in," he says, "and it was like avalanche conditions. No one was going to go anywhere. It was an amazing thing."

There probably wasn't a blizzard. Historical weather data from Dark Sky suggest there was some light snow in the days leading up to and during the retreat. But the weekend did—at least, metaphorically speaking—bury a lot of what came before it.

"I have been in many, many of these kinds of meetings throughout my career," recalls Highsmith. "This one was really special."

I spoke with 16 of the 17 attendees. (Kent Beck, a technical coach at Facebook, declined to be interviewed for this article.) Over a decade and a half later, they reflected on the retreat. "It was one of those things where you think, 'You know, you're gonna get a bunch of people in a room and they're going to chitchat and nothing's going to happen,'" says Martin. "And that's just not what happened. This group of people arranged themselves, organized themselves, and produced this manifesto. It was actually kind of amazing to watch."

Settled at Snowbird, the group began laying out what they had in common.

Schwaber recalls, "When we compared how we did our work, we were just kind of astonished at the things that were the same."

Unlike other historical documents, the Agile Manifesto doesn't declare truths self-evident. Rather, it compares: We value this over that. This construction, some of the framers say, is one of the crucial features of the document. Of course, it's unclear who came up with it, though several of the document's framers have their theories.

Ward Cunningham, the cofounder of Cunningham & Cunningham (who is famous in the software community for, among other things, coining the term "wiki"), reflects on that moment. "When it was written down on that whiteboard, some people were out in the hallway on a break," he recalls. "And I was out in the hallway, and [someone] said, 'Come here, and look at this. Look at what we wrote.' And we were just standing around looking at that whiteboard, in awe at how much that summarized what we held in common. It was such a dramatic moment, you know, that instead of everybody talking in small groups, we stood around that whiteboard and studied it."

Cunningham says he jumped on a chair and took a picture of that moment "because I could tell that something profound had happened."

So what is the Agile Manifesto? The preamble reads, "We are uncovering better ways of developing software by doing it and helping others do it." It then lays out the four core values:

- 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 document concludes that "while there is value in the items on the right, we value the items on the left more." Like any good founding document, the words can be interpreted differently, but the basic gist is this: Put people over process. Focus on making software that works, not documents about that software. Work with your client rather than fight over a contract. And along the way, be open to change.

The men finished the Manifesto during the retreat. They spent the rest of the time working on the 12 principles behind the new document and, well, skiing. (Some argue that the principles are considered an official part of the Manifesto; others consider the Manifesto itself just the values.)

This new philosophy needed a name, and not everybody was satisfied with the "Lightweight" working title. "It somehow sounds like a bunch of skinny, feebleminded, lightweight people trying to remember what day it is," Highsmith recalls Alistair Cockburn, another participant, saying. Cockburn, who is today an independent software consultant, remembers facilitating the search for the word. "It wasn't like somebody said, 'Agile! Oh great, let's go,'" Cockburn tells me. "It was really a lot of work." The other finalist, he says, was "Adaptive."

"The only concern with the term 'agile,'" writes Highsmith in his 2001 summary of the retreat, "came from Martin Fowler (a Brit for those who don't know him), who allowed that most Americans didn't know how to pronounce the word 'agile.'" Fowler eventually got over it—although, after speaking with him on the phone, I couldn't help but notice that he still pronounces the word with a British accent: an elegant *ah-gile*, instead of the American *ah-gel*.

Many described the debates at Snowbird as surprisingly friendly, especially considering the intensity of the egos in the room. Cockburn recalls "immense generosity in the listening and the understanding for other people."

But not all the participants remember everything so rosily: "The first day had quite a lot of alpha-male-type, status-posturing-type behavior," Brian Marick, an independent programmer and author, recalls, "which made me pessimistic that much good would come out of the meeting." Marick says he called his wife that first evening of the retreat, telling her, "There's a powerful odor of testosterone in this room."

Schwaber says the group did invite "a whole bunch of really pretty knowledgeable women" but that none showed. "They thought it would just be a carousing and smoking weekend," Schwaber says. "They didn't think we were going to do anything intellectual or productive."

"That was a shame," he says, "because there's some people that would've been really helpful." But it's unclear whether women were, in fact, actually invited: A few of the framers tell me they vaguely remember some women being invited. Others don't.

As the men left Snowbird, no one anticipated what happened next. "When I came down the mountain, which I was riding with a couple other Manifesto writers, I was thinking to myself, 'I'm not sure anybody would pay any attention to this,'" recalls Mike Beedle, the CEO of Enterprise Scrum. "My feel was that it was sort of like a gamble. It was like a question mark. Who knows? I mean, maybe people will go to this website that we're proposing putting up. Or maybe they won't."

* * *

They did. Unlike the ink-and-paper Declaration of Independence, the Agile Manifesto was born of the internet age. The final document is hosted online, on a simple website that feels straight out of the early 2000s, featuring a bunch of guys in khakis standing around a whiteboard. Cunningham, who continues to host the site, says he intended for people to print the document and hang it up as a poster. But for a decade and a half,

the website provided something greater than cubicle art—it stood as a virtual, communal rallying cry. Site visitors were invited to sign on to the Manifesto and publicly add their names to the document.

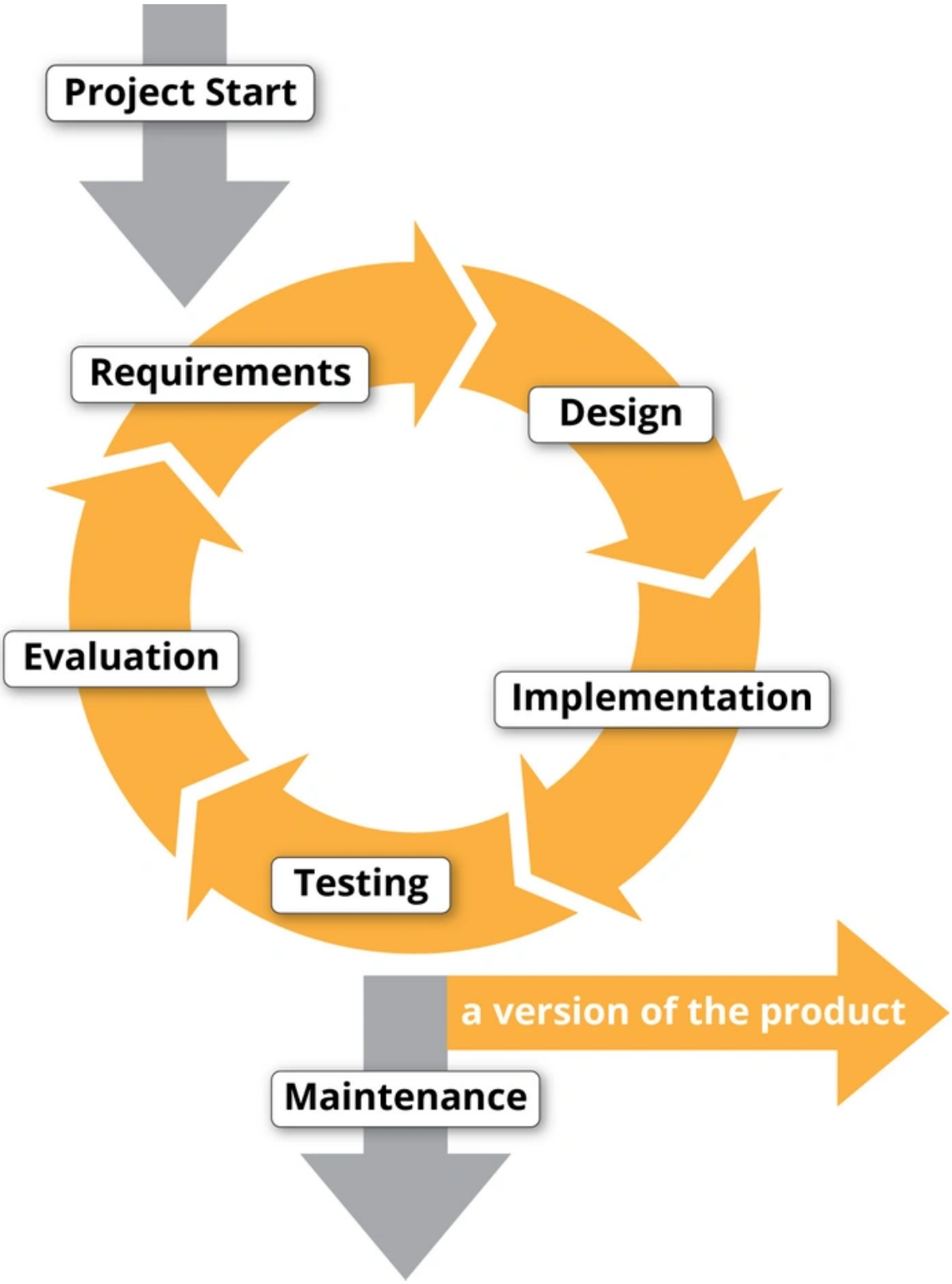
"We put that thing up, and it just exploded," says Dave "PragDave" Thomas, a coauthor of *The Pragmatic Programmer* and an adjunct professor at Southern Methodist University. "That site was actually a focal point, if you'd like, for people who want to say, 'Yes, I agree with this.' And I think that's one of the reasons it took off."

Marick agrees. "I think it was really the fact that people could vent their frustration by citing Martin Luther's theses hammered to the door, and they could put their signature on it as well," he says. "That was what really gave it momentum."

The ability to sign the document ended in July 2016. (Cunningham reconfigured the hosting and is beginning to treat the Manifesto like a historical document.) But in the 15 years since it was first published, he says, more than 20,000 people signed the Agile Manifesto.

* * *

The Manifesto, of course, was only the beginning. "My gosh, I wish I could've been there," Grady Booch tells me. Booch, the chief scientist of software engineering for IBM Research, was invited to the retreat at Snowbird, but says he bailed at the last minute in order to deal with a "pesky customer." Booch doesn't doubt Agile's seminal origin or its subsequent impact. He tells me that the 1990s were "an incredibly rich time in development in software engineering, when you had literally dozens, if not hundreds, of people that were pioneering new ideas about software development." All of that, he says, "came to a head" at Snowbird.



An Agile process (Courtesy of the Computer History Museum)

Unlike Waterfall, Agile emphasizes iterative development, or building software in pieces. Agile teams typically work in short cycles—which are

called “sprints” in Scrum, today one of the most widely used forms of Agile—that usually last two weeks each. Booch argues that both Agile and Waterfall are valid approaches, but that different projects call for different methods—and it’s important to weigh factors like the project’s risk and the culture of the team that’s executing. “If I’m building a nuclear-power plant,” he says, “believe me, I don’t want to use incremental and iterative methods because testing failure is never a good thing; it’s kind of irreversible. On the other hand, if I’m building a throwaway app for some new clone of Tinder for goats, whatever it might be, then sure, I’m gonna put a few people in a room and go build this.”

Perhaps Agile, or something like it, was inevitable: If software projects were going to be successful in the nimble, digital-first future, they needed to be able to, as goes the tech parlance, pivot—to respond to changes. The web profoundly changed the way software is delivered. Today’s software isn’t typically burned onto a CD-ROM and stocked on a store shelf; updates can be pushed to your laptop or smartphone remotely. This makes it easier to add features or fix bugs after releasing the product.

“In order to succeed in the new economy,” Highsmith writes in his 2001 summary of the retreat, “to move aggressively into the era of e-business, e-commerce, and the web, companies have to rid themselves of their Dilbert manifestations of make-work and arcane policies. This freedom from the inanities of corporate life attracts proponents of Agile Methodologies, and scares the bejeebers (you can’t use the word ‘shit’ in a professional paper) out of traditionalists.”

But this isn’t just a software story. Today, teams across industries and around the world are “going Agile”—or, at least, using bits and pieces of the Agile philosophy. The document itself has been translated into over 60 different languages.

Cockburn believes that’s because what Agile “managed to decode was

something about pure mental, team-based activities"—and that "it's just an accident of history that it was the programmers who decoded this."

Compared with "the more mainstream, more Waterfall-ish kind of ideas" that "lay out in great detail what everybody does," Agile is "much more empowering to the individuals doing the work," Fowler says. And, since it has been adopted by a spectrum of professions, Arie van Bennekum goes so far as to suggest changing the word "software" to "solutions," in order to open up Agile to everyone.

Despite discussions over whether the Manifesto itself should be amended, many of the original signers see the document as a historical—not a living—document. "It's like a Declaration of Independence in U.S. history," says Cockburn. "You don't go back and rewrite that."

"I think those four bullet points are still as valid as ever," says Grenning. "I don't expect them to change."

With the end of public signing, it seems unlikely that the Agile Manifesto will ever officially change, but that doesn't mean there aren't problems in the world of Agile. Over the course of our conversations, many of the framers expressed a frustration with modern Agile. On the heels of Agile the philosophy came Agile the industry: Agile software, Agile coaching, Agile trainings, and Agile conferences. There's no shortage of ways you can spend money to try and make your business or team "Agile."

But there's a particular irony here: Agile is a philosophy, not a set of business practices. The four bullets outline a way of thinking, a framework for prioritizing all the complicated parts of a project. They don't tell you what software to buy or how to hold your daily team meeting. "Now you can go to a conference, and there's aisle after aisle of people who are selling you computer tools to run your process. And they say it's Agile," says Cunningham. He points to the first value of the Agile Manifesto. "It says, 'Individuals and interactions over processes and tools.' How did [Agile]

become a process-and-tools business?"

Cunningham thinks that "other people saw dollar signs and wanted to do the dollar-sign thing." He adds, "Money has been made."

Van Bennekum, who is today a thought leader at Wemanity, says, "I see people being an Agile coach absolutely not knowing what they're talking about," which is "upsetting."

Meanwhile, Jon Kern, the chief architect at Fire Planning Associates, admits he "kind of stepped out of the Agile ring"—exhausted after a lot of people just didn't get it. "You get a lot of people, just snake-oil salesmen—folks that say they're doing Agile when it's Agile in name only," he says. Kern compares Agile to yoga, arguing his practice is personal and that he doesn't "try to tell other people how to practice."

The monetization of Agile aside, the influx of nontechnical users has created some conflict. Martin maintains that the "most annoying aspect right now" is that Agile "has been taken over by the project-management people," leaving "the technical people and the technical ideas" behind.

Jeff Sutherland, a cocreator of Scrum and the CEO of Scrum, Inc., is frustrated by misreadings of the document within the software community itself. Sutherland says he sees teams in Silicon Valley that claim to be Agile, but are "not delivering working product at the end of a short iteration." This, he says, puts them "in violation of the second value" of the Manifesto. "This kind of thing that most people are doing that they can't get anything working in any reasonable time—that they claim is Agile because anybody can do whatever they want—is not consistent with the Agile Manifesto," he points out.

A few have gone so far as to proclaim that Agile is dead. But Cockburn argues that there's always some benefit to trying Agile, even if it's not perfect: "Even badly done, Agile outperforms all of the alternatives, or at

least, the main older alternative [Waterfall], which is out there."

The intricacies—and passions—of these debates demonstrate just how big Agile has become. When I ask Kern what expectations he had leaving Snowbird all those years ago, he laughs. "For the 10th-[anniversary] reunion, we were trying to come up with some words to put on a T-shirt," he explains, "and my proposal was: 'Four measly bullets, and all this shit happened.'"

We want to hear what you think about this article. Submit a letter to the editor or write to letters@theatlantic.com.