

## Will the Build Bottleneck Put the Brakes on Agile?

People are more likely to gum things up, developers say

BY GEOFF KOCH

For those keeping track, the Agile Manifesto turned 4 years old in February—a moderately mature age in the fast-changing software world. According to several developers, consultants and software executives, the manifesto's proposed lightweight alternatives to process-heavy Dilbertesque coding methodologies are increasingly used across the software development spectrum.

"It sure feels like [agile methodologies] are gaining traction to me—in a big way, in fact," said Tim Walker, a development manager at Boulder, Colo.-based Webroot Software, a company focused on privacy and anti-spyware products.

But as agile moves from hobbyist and hacker to the Fortune 500, critical challenges await.

A handful of software vendors and consultants see a looming build management bottleneck, as complex development organizations start reaching for the agile ideal of automated continuous integration.

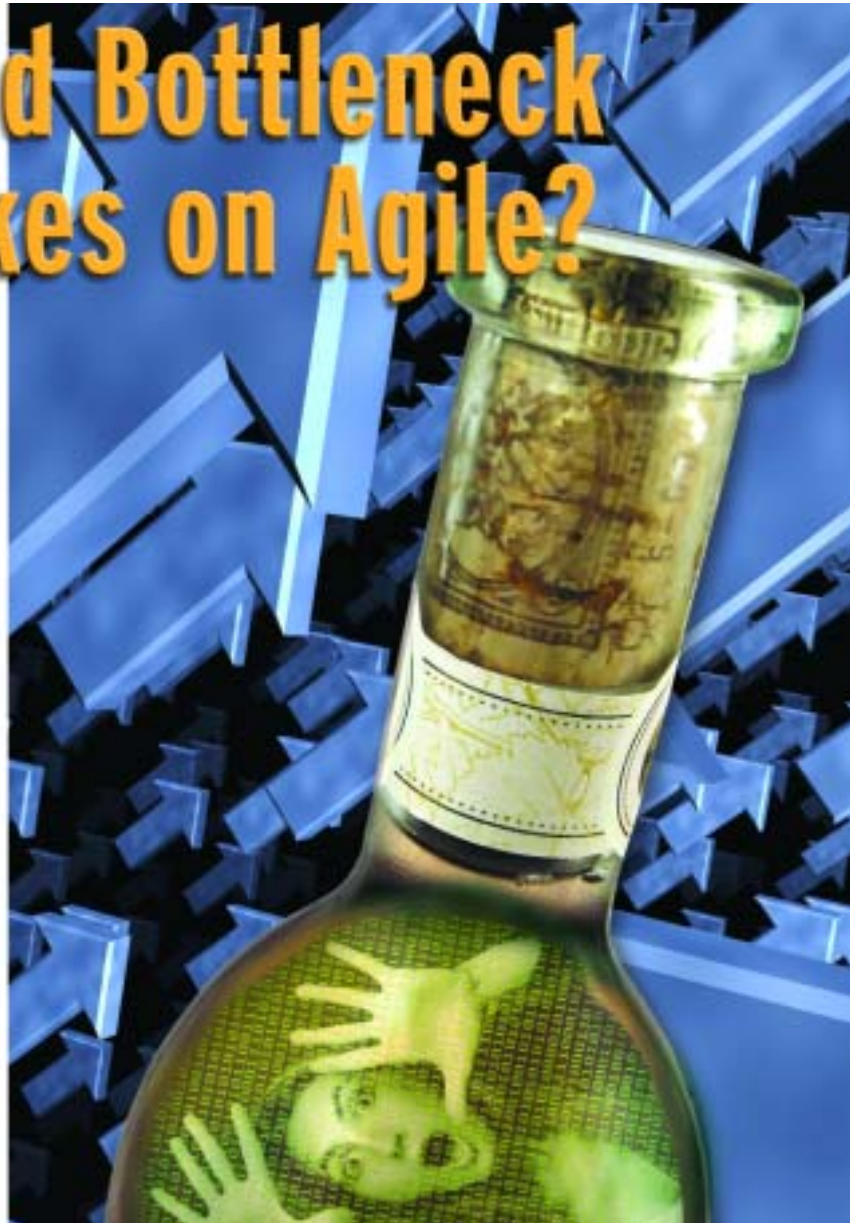
Building is one thing for the solo developer who just hits the F12 key to compile at every cof-

fee break instead of every few days.

It's quite another for a 500-person development organization working across geographies; coding in C++, .NET, Perl and Python; updating 20 different products; targeting a handful of different target platforms; and, of course, being driven to hit increasingly compressed deadlines.

Other developers say the bottleneck blame is more appropriately spread across different parts of the coding cycle, or more commonly, when it's assigned to the most error-prone part of any process: people.

Whichever camp is right, digging beneath the agile and build management foundation is a sure way to unearth opinions on everything from the virtues of version control to the proper use of lava lamps in software development. (Hint: tell your spouse not to throw it away, yet.)



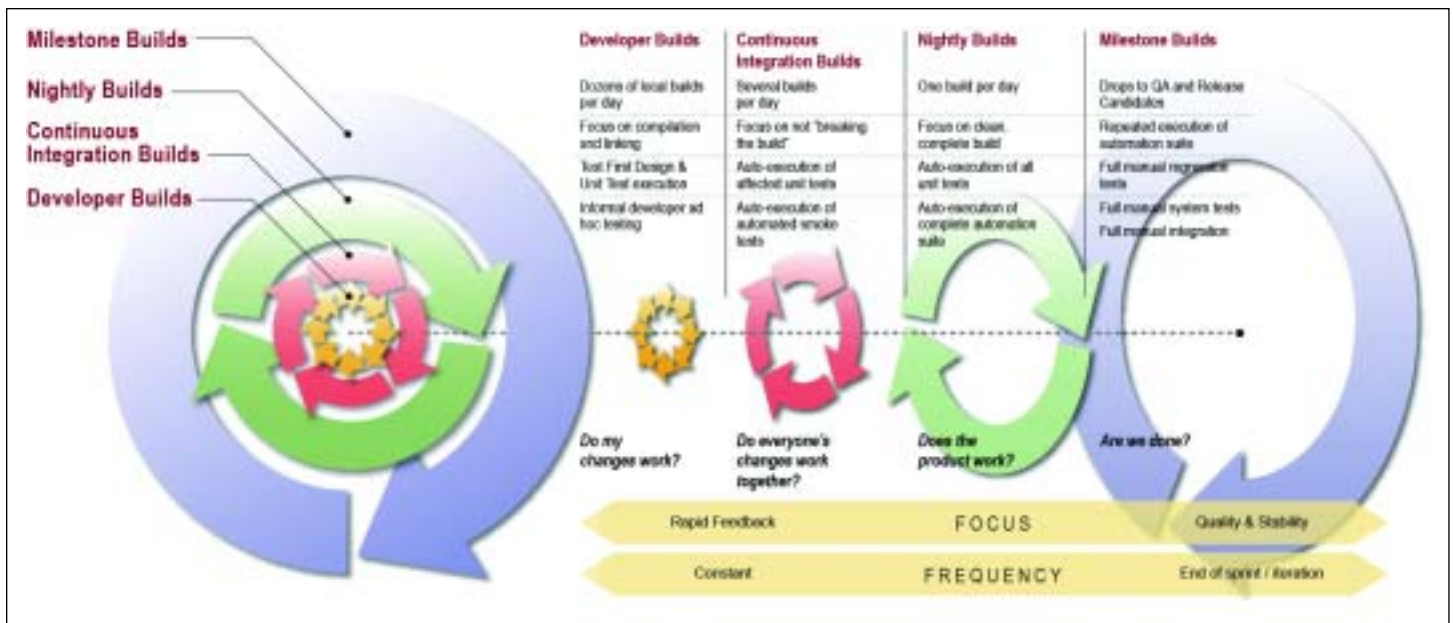
### VIEWING MARKET POTENTIAL

Years ago, Joe Senner wrote code for and maintained part of the Unix kernel. As he remembers it, builds took an entire weekend and rarely went smoothly.

"Streamlining the build process is a quality-of-life issue for developers," said Senner, today the chief technology offi-

cer of BuildForge, an agile-oriented application life-cycle management company he founded in 2001.

BuildForge has successfully slashed build times at several leading software companies. According to Senner, his ALM tool helped Electronic Arts to reduce a 60-hour build to a three-hour build, Adobe Systems to increase the number of



code-build-test cycles from 18 to 360 per week, and telecommunications equipment maker Avaya to turn a 10-hour build into an unattended 10-minute pushbutton process.

To evaluate eye-catching claims like this, it's important to weigh plain old process improvement against the benefit of the tools themselves, said Mike Clark, an agile-focused programmer, consultant and author.

"Teams that want to be more agile are headed for a train wreck if they have long build times; they'll need to find ways to build all or part of the software more frequently to get the kind of continuous feedback that helps agile teams move quickly," said Clark, who hasn't used BuildForge. "But I don't see that as a tool problem as much as it's a build process problem. That is, I don't think tools will help as much as getting someone on the team to optimize the build process will."

Optimization may be easier said than done. New languages and libraries ooze increasing amounts of code while the 20-year-old Make-based infrastructure lags behind advanced IDEs. Worse, ongoing coding projects create deeply recursive and dependency-laden groups of Makefiles that only a conjurer can untangle.

According to software build

infrastructure company Electric Cloud, current approaches to dealing with this complexity and speeding associated build times aren't working. Distributing builds across multiple processors in an SMP machine is too expensive, manually partitioning Makefiles is too difficult and time-intensive, and doing incremental builds for production software (instead sending complete builds through QA on the way to release) is too risky, the company says.

"We have found the build to be a bottleneck in most software development organizations, and our studies have shown that, on average, developers spend between 10 percent and 20 percent of their time waiting for builds," said John Ousterhout, founder and CEO of Electric Cloud. "The problem has been that organizations don't understand that the bottleneck can be easily eliminated, so they have gotten used to walking with a limp."

Electric Cloud's growing batch of case studies describe several customers—Force10 Networks, Intuit and Qualcomm—using the company's tools to streamline their build process and implement continuous integration.

#### 'NOT ROCKET SCIENCE'

While they're not necessarily

described in slick case studies, several open-source continuous integration tools exist. And many agile adherents say these community-built alternatives more than suffice.

"A build tool is not rocket science," said Robert "Uncle Bob" Martin, CEO of software consulting company Object Mentor and co-author of the Agile Manifesto ([www.agilemanifesto.org](http://www.agilemanifesto.org)). "Any team that can build a software project can build a tool to build that project. The tool does not need to be complicated, nor does it need to solve world peace. All it has to do is build the system, run the tests and report status."

Open-source and agile-friendly tools include Ant and Make for automatic builds; Anthill and CruiseControl for continuous integration; and FIT and JUnit for testing.

Even when it comes to integrated development environments—tools that many say are worth their complexity—open-source alternatives exist.

Two of the leading Java IDEs, Eclipse and IntelliJ, are available in open source. According to Martin, both IDEs are tightly bound with agile processes, including unit and acceptance testing and support for refactoring. (ReSharper, a VisualStudio .NET add-in, may

make for better agile programming in .NET environments, though it's not open-source.)

Martin said that FitNesse, a software development collaboration tool, closes the last major gap in the agile tool suite—how customers tell programmers what the software should do. FitNesse, intended in part to allow customers to specify requirements without having any specific coding knowledge, should complement JUnit, an open-source Java testing framework used to write and run repeatable unit tests.

James Shore, a Portland, Ore.-based Extreme Programming (XP) and lean development consultant, said the do-it-yourself approach is consistent with the core principles of agile programming.

"In the early days of agile, people asked how they could pair program when they worked in cramped cubicles," Shore said. "The response from the agile adherents was to grab a screwdriver, take down the cubicle wall and make room for another person. It's always been more about empowering people than worrying about technological control."

BuildForge's Senner doesn't seem worried about countering the open-source claims. Eventually, organizations realize that a proprietary build system is not their

# YOU CAN'T DO AGILE DEVELOPMENT WITH SLOW BUILDS, PERIOD.

**ELECTRIC CLOUD** eliminates the build bottleneck and enables agile development. Electric Cloud transforms inexpensive servers into highly scalable clusters that speed up builds 10 to 20x. Leading companies such as Intuit, Qualcomm, Agilent, and Expedia have eliminated their build bottleneck with Electric Cloud.

*Break the  
Build Bottleneck*

electric  cloud

[www.electric-cloud.com](http://www.electric-cloud.com)



core competency, similar to the way they realized this about databases, source-control systems and defect-tracking systems, he said.

“Essentially, the open-source tools can accelerate builds from one to three times, but they fall over dead after that,” said Ousterhout, adding that most of Electric Cloud’s customers first failed with open-source tools. “The reason this happens is that builds have hidden dependencies that cannot be expressed in the Makefiles.”

#### A BUILD SOLAR SYSTEM

Asking whether build is an agile bottleneck may be too simplistic a question, given the nested complexity of many development teams.

James Van Riper, vice president of research and development at change management company Serena Software, manages one such team. Van Riper’s 60 developers work out of their homes and offices across the United States and the United Kingdom.

Serena recently transitioned from milestone builds to continuous builds. To help think through the transition, Van Riper built a model roughly patterned on the solar system. The concentric circles, or orbits, represent build iterations.

“The closer you are to the center, the faster you go and the more builds you do,” Van Riper explained.

At the center of Van Riper’s solar system is the single developer doing regular and frequent builds on his desktop while working on his particular coding assignment. The developer likely runs a short list of unit tests as part of this iteration.

The next ring is where continuous integration happens, perhaps for a subteam or maybe for the whole group. The timer can be set to kick off the build every 15 minutes, including all the code that’s checked in at that point along with a larger group of tests, including a smoke test.

Then there’s the nightly build, which includes the entire codebase. On the outer ring is the milestone build that is handed off to QA after passing the complete suite of tests.

What keeps these development worlds turning?

“As a company, we like to mix it up so we can experience what our customers are experiencing,” Van Riper said. “For the most part, we’re using Anthill. But for our sophisticated processes, we end up adding a lot of stuff. It isn’t install it and go.”

Van Riper’s diagram at least hints at all the human glue that holds together even the most automated agile shops. The best-laid agile plans can go awry if people don’t use the tools appropriately.

Lazy use of version control, intentional or otherwise, is one such pitfall.

“Even if builds are happening with some regularity, if all programmers aren’t checking in their updated code frequently, then bugs start sprouting and it becomes difficult to back out and fix bad code,” said Richard Leavitt, vice president of Boulder, Colo.-based Rally Software Development, which provides coaching and on-demand tooling for scaling agile software development.

And that assumes a version-control system exists at all.

Andy Hunt, programmer, author and another co-author of the Agile Manifesto, said that many development shops in the United States still don’t use any form of version control and instead simply build the product using the last settings in the IDE. Changes easily can get lost, and lots of time is wasted tracking down bug artifacts of the ad hoc build process.

“Some widely used version-control systems are well known to corrupt and lose source files on occasion,” Hunt said. “That’s just incredible to me, especially since excellent free, widely used, open-source solutions abound—CVS and SVN come to mind.”

Even some who are otherwise fans of open source cite the importance of good version control.

Rajiv Delwadia, chief technologist at life-cycle planning and management application company VersionOne in Atlanta, said that most open-source tools are good enough for most teams. “But if you’re going to pay for anything, pay for good source control,” he added. (Delwadia’s team uses Microsoft’s Visual SourceSafe.)

#### THE HUMAN DIMENSION

Whether the build process puts the brakes on agile and when to consider proprietary tools misses the point for software consultant and trainer Mary Poppendieck.

“Absolutely, tools and technology are not the [agile] bottlenecks,” she said. Instead, Poppendieck, the author of “Lean Software Development: An Agile Toolkit,” points squarely at people-centric problems:

- Approval processes that dump large batches of work on a development organization at once, or that overload the development organization.

- Detailed and ongoing customer clarifications of what they really want (particularly when they don’t really know).

- Testing that occurs long after development so developers don’t know if they are doing the right thing or whether tons of work has to be redone.

- Deployment—especially when code is tossed over the wall at users and support organizations that have not been involved up to that point.

A narrow focus on the agile build-management tension also may obscure the big-picture view of agile’s slow ascendancy as a mainstream software methodology. And it certainly misses the fun developers are having as they integrate often and stay close to shippable code.

Clark has heard of organizations setting up Web pages with RSS feeds about the output of the continuous build so that real-time information is constantly pushed to the person who needs it. For those who want to trade RSS for more retro-technology, Clark’s blog has an example of setting a lava lamp to a continuous build machine. The lamp, powered by X-10 equipment, glows green if the build is OK, red if it’s not.

“Continuous integration offers all sorts of fun ways to get feedback on the health of software,” Clark said.

Software development as fun? Just don’t tell Scott Adams. This could finally cause new Dilbert material to dry up. ■



2307 Leghorn Street, Mountain View, CA 94043  
Phone: 650.968.2950  
Fax: 650.968.6000