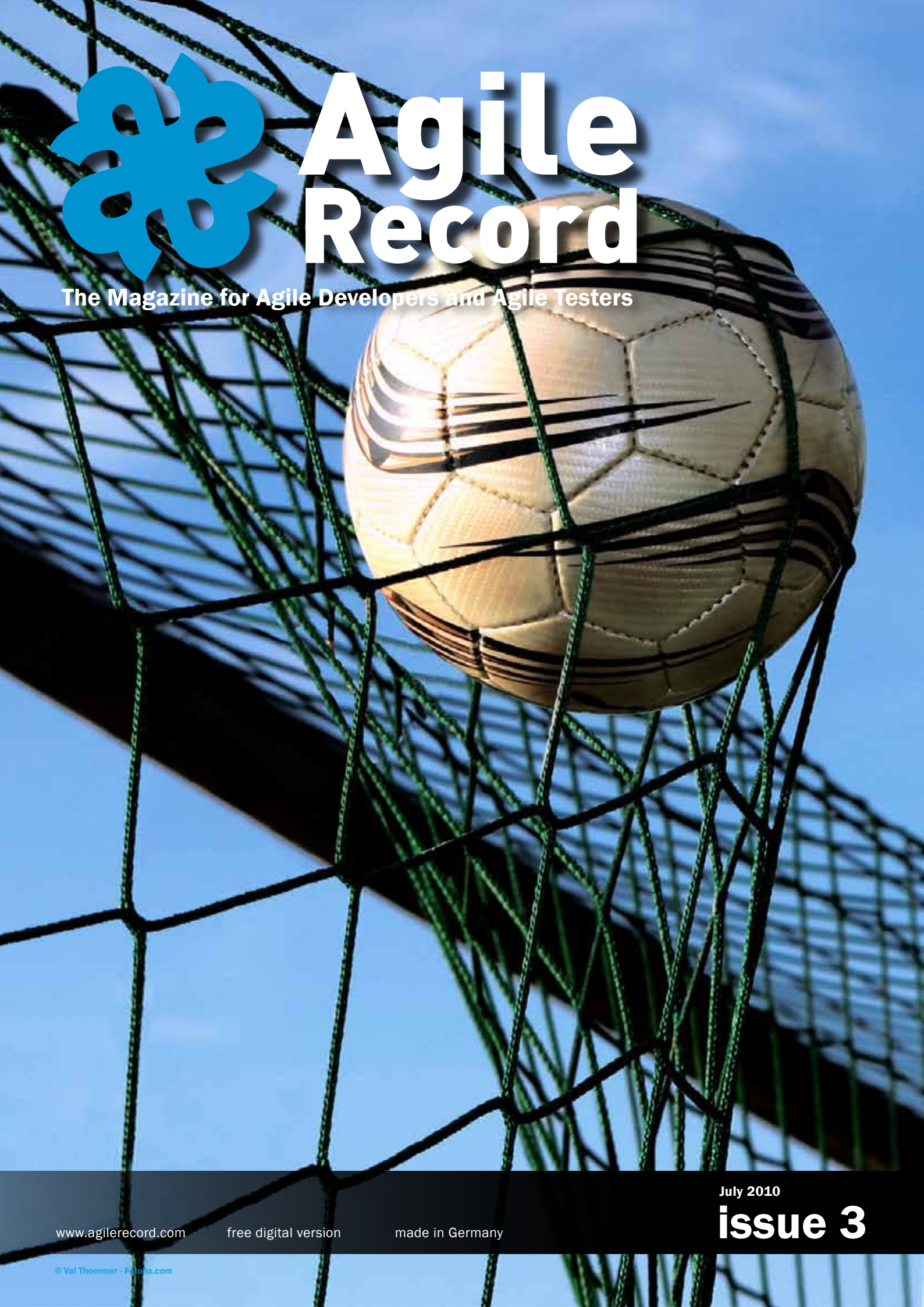

Agile Record

The Magazine for Agile Developers and Agile Testers

July 2010
issue 3

www.agilerecord.com free digital version made in Germany

What Really Happened

by Catherine Powell

Graphs and numbers look like they contain a lot of information, but they can be misleading. This is the story behind the chart. This is the number of found and fixed defects during an iteration, and more importantly, this is what really happened.

The Chart

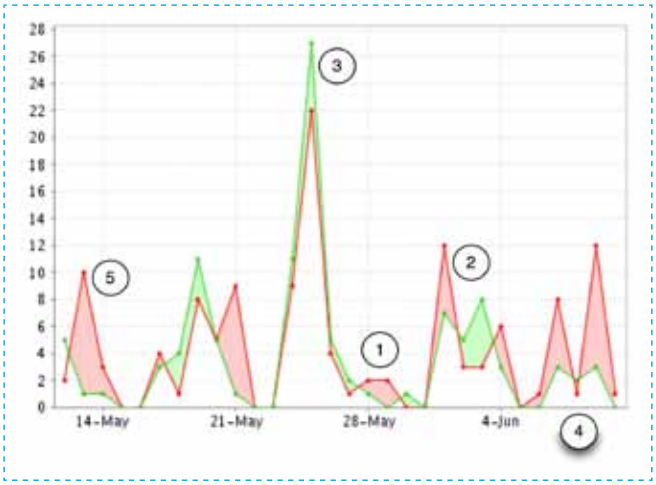

Figure 1

This chart shows the number of defects found (red) and the number of defects resolved (green) over a thirty-day period on a single project. This is the middle of a project, with new features being coded and checked in, ongoing tests being conducted, and defects being identified, fixed, and verified.

What It Looks Like

On the surface, this chart tells a story. It says that bugs are getting found, and bugs are getting fixed in about equal measure. The code is churning, which is normal for the middle of a development cycle, but isn't trending toward stability. The number of bugs found slightly exceeds the number of bugs being fixed, so this is a project to keep an eye on. The developers (and testers) may need some intervention!

But is that really the right story?

What Really Happened

Background

This project is a software solution being developed by a Boston-based team. The software is a library that runs on several Linux variants and provides services to enterprise OEM customers. The team relies heavily on automation for testing, and has a large lab that runs developer tests, unit tests, build verification tests, nightly tests, and multi-node scaling tests near constantly. Defects are logged by machines (automatically with scripts) and by humans. Any test that does not complete successfully is considered a failure; tests that do not run or that are faulty are logged as defects, as well as test failures. In addition, support engineers log defects based on customer escalations.

Episode 1: Memorial Day

The Memorial Day holiday occurred on May 31, 2010 in the United States. The office was closed, and little to no work was done. The drop in the number of defects identified and in the number of defects fixed has nothing to do with the software. Rather, it's because no humans were finding or fixing defects.

That apparent "increased stability"? That was really a barbecue and a team ignoring their computers for a long weekend.

Episode 2: The Heat Wave

In the first week of June 2010, Boston experienced a heat wave and high humidity. It's not uncommon, but this happened earlier in the summer than normal. The building was unprepared, and the air conditioners were not yet turned on for summer weather. The spike in defects is the lab getting warm. Several machines simply got too hot and died. When they died, the tests that hap-

pened to be using those machines failed with an error and automatically logged a defect.

So a spike in heat led to a spike in defects created. There is a spike in resolved defects just afterwards, as the team replaced the dead machines and tracked the test failures to those machines.

Episode 3: The File Server Update

All test machines in the lab use a central file server for storing logs and other shared information. On May 24, the IT team deployed a new file server. This would have been fine had no tests been running, but one set of tests was still going. When the central file server went down (for replacement), it exposed a bug in the test infrastructure. The test infrastructure panicked, discarding machines that didn't work and grabbed new machines, over and over. This stopped when the central file server came back up, by which time the test infrastructure had used and discarded 24 machines.

One bug in the test infrastructure, and one triggering event in the form of file server downtime caused 24 defects to be logged. The fix took 10 minutes to implement.

Episode 4: The New Operating System

This project is software running on various flavors of Linux. In general, the various flavors of Linux are roughly similar, but they are each quirky in their own way. In mid-June, the team decided to start supporting a new type of Linux: Red Hat Enterprise Linux. A team member installed Red Hat Enterprise on a few machines, put them in the lab, and configured the tests to start running on those machines. That night, some of the tests failed, and the team came in the next day to find some bugs specific to Red Hat Enterprise Linux. Some were fixed, and another spike occurred the next day as follow-on issues showed up.

This spike in defects logged is the result of a new feature of the software – support for a new operating system.

Episode 5: The Vacation

In mid-May one of the developers took a week of vacation. With a small team like this, the developer's vacation meant that 20% of the team was gone for a week. The chart shows an unusually low number of defects fixed; this is because a significant portion of the team simply wasn't around to fix things. The following week – when the developer returned from vacation – shows a jump in the number of defects fixed.

Is that indicative of a problem? No. It's indicative of a vacation.

Conclusion

The chart shown above implies a story. At first glance it's a story of some churning code and some developers who just can't keep up. That story is wrong. A look at what really happened points to two things: a small team, and an unstable test lab. A more "spikey" chart is a consequence of a small team, where the effects of one team member being gone are highly visible; on larger teams the effect of any one person is generally smaller, so charts are smoother. For the manager looking at this chart, understanding the story behind the lines of the chart points to needing to spend more effort on stabilizing the test lab rather than on changing a development team's behavior. ■

> About the author

Catherine Powell
has been testing and managing for about ten years. She is a manager, a tester, an author and a formal mentor to testers and test managers. Catherine has worked with a broad range of software, including an enterprise storage system, a web-based healthcare system, data synchronization applications on mobile devices, and web apps of various flavors.

With a focus on the human side of software development, Catherine builds strong teams spanning testers, developers, support, product management, and all the people involved in turning an idea into reality. She emphasizes the generation of information and pragmatic decision making using a myriad of approaches. With thoughtful techniques that access the strengths of the human team members combined with the needs of the system and its users, Catherine guides both developers and testers to be a valuable part of the decision making required to create rock-solid software.

Catherine focuses primarily on the realities of shipping software in small and mid-size companies. Specifically, she highlights and works to explicate the "on-the-ground" pragmatism necessary for an engineering team to work effectively with both software and humans from product definition through release, and in the field.