Using Analytics to Guide Improvement during an Agile–DevOps Transformation

Barry Snyder, Fannie Mae

Bill Curtis, CAST

// Fannie Mae IT has transformed from a waterfall organization to a lean culture enabled by agile methods and DevOps. The article discusses how analytics were used, along with challenges in selecting measures and implementing analytics in an agile–DevOps transformation. //

SOFTWARE ANALYTICS[1] HAVE proven increasingly useful in industrial case studies for guiding adoption and evaluating benefits of advanced software techniques.[2,3] In 2015, Fannie Mae, a provider of liquidity for mortgages in the US housing market, undertook a major transformation from its traditional waterfall approach to a culture built on agile methods and DevOps practices.[4]

The transformation was motivated by Fannie Mae's need to adjust its IT capabilities to meet the requirements and pace of a rapidly evolving mortgage market. In 2012, Fannie Mae had only 10 teams using agile-like techniques. Releases took nine to 18 months, provisioning of environments took two to four months, and quality was not consistently measured.

Fannie Mae's IT department was a complex, technical-polyglot ecosystem composed of 461 applications, several hundred utilities, and almost 18,000 open source components. As part of reducing this complexity, Fannie Mae initiated two projects to implement agile methods and DevOps as enterprise capabilities. Executive management required periodic empirical evaluation of progress in quality, productivity, and delivery speed to monitor the effectiveness of the agile–DevOps deployment.

Implementing the Analytics Platform

Fannie Mae started by leveraging what already existed and filled the gaps with new solutions. While this got the transformation moving forward, it revealed that a highly customized solution can become an impediment. For example, the production release platform was built out of three point-solution products with significant custom scripts to unify these solutions. While it conformed to Fannie Mae's tightly governed environment, it impeded transformation because a major update to the platform would take more than nine months when team needs were in the weeks.

In comparison, the CAST Application Intelligence Platform

(AIP)—adopted for application analysis and measurement—had no customization and was implemented out of the box. This platform was deployed in 2014 as an enterprise platform providing technology-independent structural-quality metrics that enabled consistent analysis and comparison across Fannie Mae's application portfolio. Structural quality represents how well the system is architected and the code is engineered. Since manual structural-quality analysis of entire applications was infeasible, an automated platform provided a common analytics solution across the enterprise. The only customization automated the AIP onboarding process with an in-house built framework—a necessity for providing a self-service capability that was integrated into the DevOps toolchain.

AIP analyzes an entire multilayer, multilanguage application system from the user interface to the database structure.[5] To evaluate architecture, structural-quality analytics are developed during integration rather than during unit testing, and replace some information previously provided by formal inspections. An abstract structural representation of the system is generated from this analysis and evaluated against more than 1,200 rules of good architectural and coding practice. Violations of these rules are weighted and aggregated into measures that highlight the operational and cost risks of the application system in five areas: robustness, security, performance efficiency, changeability, and transferability (or comprehensibility). These five measures are aggregated into a sixth measure—the *total quality index* (TQI)—that provides a summary structural-quality score.

Originally, analytics were delivered by a central team, so development teams were constrained in using measures by how quickly they were received. This bottleneck led to developing a self-service interface enabling teams to choose the frequency of scanning their applications and consume analytics at their own pace. This interface increased platform usage by 481%, from 340 scans during the year prior to the self-service interface, to 1,636 scans during the year after deployment.

Analyzing Productivity

Measuring productivity improvement was a critical challenge in monitoring the transformation. Several measures (story points, dollar spend, LOC, code review defects, code coverage, etc.) were evaluated and eliminated because they did not provide equivalent comparisons that were independent of technology and comparable across applications. For analyzing productivity, Fannie Mae chose to measure functional size rather than LOC. The Consortium for IT Software Quality (CISQ) recently produced a standard, through the Object Management Group, for Automated Function Points (AFPs)[6] that was based on counting guidelines published by the International Function Point Users Group (IFPUG).[7] Since AFPs were automatically generated by AIP when analyzing the structural attributes of a code base, AFPs experienced rapid acceptance by early adopter teams.

AFPs provided an outcome-based evaluation of productivity regarding the amount of functionality delivered per unit of effort in sprints. After aggregating results across all projects, the enterprise could rapidly adjust activities to address progress shortcomings and support transformation accelerators. In turn, teams could rapidly pinpoint which process and technology improvements drove the greatest impact.

For example, one of three squads working on elements of the same product line had adopted behavior-driven-development techniques using Selenium for testing. Analyzing data on effort, size, and development context combined from different sources indicated that the squad's productivity rose by 4.9% over three sprints due to shortening the customer feedback cycle. This led the remaining squads to adopt Selenium.

Analyzing Structural Quality

Structural-quality measures were joined with defect ratio, dollar spend, cycle release time, build count, and other measures to analyze the impact of agile–DevOps transformation practices. These analyses provided equivalent comparisons regardless of language, analyzed the complexities of multilayer transactions, and could be combined with other measures to calculate ratios such as spend per AFP, defect ratio per AFP, etc.

Application teams scanned builds one or more times during a sprint to detect structural-quality flaws and produce the analytic measures. Scan outputs provided early detection of structural flaws, enabling teams to address the most critical issues before release. Prior to analyzing structural quality, teams would occasionally discover these flaws in testing, but most often after an application was released into production. Most teams began scanning at a minimum of once per sprint (every two weeks). Those who were scanning several times a week or even daily could fix critical defects within a day or two rather than waiting until the next sprint. These measures, along with detailed information about the location and severity of defects, were provided to application teams via engineering dashboards.

Application teams used these measures to iteratively improve quality in the five structural-quality dimensions while monitoring overall quality via the TQI score. Structural-quality analytics enabled teams to look for code quality issues presenting short-term risk, while simultaneously addressing long-term architectural limitations and costly maintenance early in construction. Fannie Mae intended to improve the structural quality of all applications to a "green" state—roughly a 70th percentile score on the TQI. At the start of the journey in 2015, most apps were "yellow" and "red" (scores averaging the 46th percentile on TQI). Currently, nearly all applications have achieved a green state.

Aligning Analytics across the Organization

Structural-quality analytics provided a multidimensional, enterprise view of the code base via

- a visual heat map—the red-to-green map of applications for both quality and size;
- enterprise trending of scores—a 2D line chart of overall code quality over time for all applications presented by portfolio, subportfolio, or application;
- aggregated structural flaws across the enterprise—the top 10 quality issues across all applications; and
- filters based on portfolio and maturity of transformation—filtering indicating which applications were agile versus waterfall or DevOps versus non-DevOps.

However, the structural-quality output did not align with other enterprise metrics. Aligning the metrics at the enterprise level and across the DevOps tools was a challenge, as the tools were in silos. As the transformation evolved over the first two years, each platform built one-off solutions to collate its data with external enterprise metrics—this resulted in metric silos.

To eliminate the silos, Fannie Mae constructed an enterprise data mart to align measures and analytics from all enterprise tools covering financials, enterprise program management, software development (DevOps, test automation, test data management, agile methods, code quality, etc.), and operations. The collation of data across platforms enabled holistic insight into the transformation progress. Additionally, relationships between data elements could be investigated rapidly.

For example, one application had a production outage identified as potential defective code. By relating the incident ticket to data from the scan prior to the production release, the team discovered it had opted to mark a reported defect as a false positive. This granular analysis enabled the team to adjust its code development practices regarding a finding flagged as a false positive.

This specific example reflects how teams leveraged the insight provided by an enterprise data mart to address specific needs. From an enterprise perspective, quality metrics could now be aligned with operational tickets, finances, and program management. This alignment enabled measuring productivity from a value (AFPs per dollar) and quality (defects per AFP) perspective. In turn, the enterprise could measure productivity and quality efficiency as trailing indicators of transformation success to demonstrate return on investment.

Analyzing Transformation Improvement

Figure 1 displays how Fannie Mae measured and monitored productivity in functionality delivered (AFPs added, modified, or deleted) and quality (critical defects per AFP) as the transformation was deployed across applications. Critical defects were those rated from 7 to 9 on a 9-point scale. During the first quarter of 2015, data aggregated across applications indicated that the transformation began delivering substantially more functionality without a decrease in quality. Applications achieved between a 30% to 48% improvement in quality when measured by critical defects per AFP compared to the 2014 baseline. An average of 28% productivity gains was achieved by teams that implemented repeatable agile practices, automated structural-quality analysis, test automation, and consistent use of the DevOps platform for continuous integration and delivery.

The agile–DevOps transformation allowed Fannie Mae to create an internal software supply chain with automated handoffs between functions to reduce delays. In comparison to 2012, Fannie Mae achieved 21 times (more than 19,000) more builds per month with half the previous staffing, while improving quality.

Using Analytic Baselines for Teams

The structural-quality analytics aggregated across applications provided a baseline for evaluating the effectiveness of each application

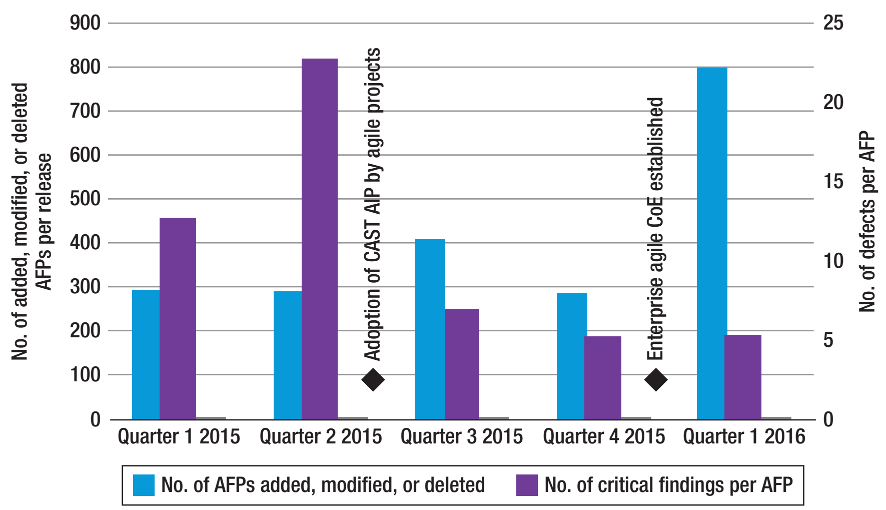

FIGURE 1. Improvements in productivity and quality during the agile–DevOps transformation. AFP = Automated Function Points, CAST AIP = CAST Application Intelligence Platform, and CoE = Center of Excellence.

team's adoption of technologies and practices. Aligning the code quality platform with the enterprise data mart enabled a correlation between the quality increases and the agile adoption of practices such as team product ownership, MVP-driven sprints (an MVP is a *m*inimally *v*iable production-ready *p*roduct that provides value to customers), test automation, and daily continuous integration. Since these practices were adopted simultaneously, it was not possible to correlate individual changes with the measured improvements. However, Fannie Mae established an enterprise-level relationship between overall structural quality measured by the TQI and the completeness of the practices being adopted:

- Teams not practicing agile methods or DevOps and conducting only infrequent scans averaged a 9.8% improvement in TQI.
- Teams performing frequent scans coupled with the continuous integration–deployment platform averaged a 24% improvement in TQI.
- Teams performing frequent scans coupled with the full spectrum of agile–DevOps technologies and practices averaged a 30% to 48% improvement in TQI, with outliers as high as 70%.

Teams began using data to evaluate the effectiveness of their agile–DevOps practices.[8] For example, one project had abandoned the use of the prebuild static-analysis tool SonarQube[9] in favor of using AIP. The project's TQI score dropped from 18% to 11% because more component-level defects escaped detection and entered builds. In comparison, projects using both SonarQube and AIP as complementary prebuild and postbuild tools averaged 24.7% structural-quality improvements measured by TQI. Consequently, the project returned to using SonarQube and AIP in tandem.

An Application Team Example

Figure 2 displays the structural-quality results for an application team that underwent the agile–DevOps transformation between 2015 and 2016. Prior to adopting agile–DevOps, the application release cycle was lengthy. The team achieved only a 10% improvement in application quality, with no means to analyze the drivers of improvement.

When the team began using structural-quality analytics, it could measure and address defect injection

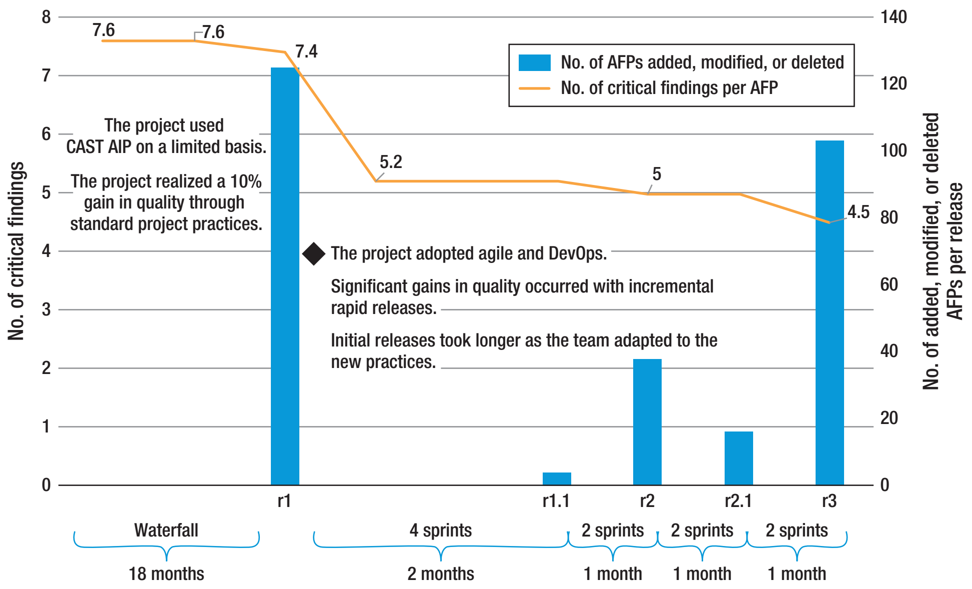

FIGURE 2. A case study of the transition from waterfall to agile practices with DevOps automation.

much earlier during the shorter sprints. The team realized a 41% reduction in defects per AFP by release r3 at the end of 2016. Coupled with this quality improvement, the team delivered more functionality (as measured by AFPs) in shorter time periods. That is, the aggregate functionality delivered in one- to two-month sprints during the first six months of agile–DevOps adoption (r1.1 to r3) was greater than the functionality delivered in the previous 18-month release (r1), resulting in a 30% improvement in productivity.

Several key lessons were learned through this agile–DevOps transformation:

- Early gains can be achieved by leveraging existing platforms, but only if those platforms can be easily adapted to future evolving needs.
- When transforming at scale, measures need to be consistent across all applications to maximize comparability.
- Measures used early in the transformation will be replaced with metrics better aligned across the enterprise as practices evolve.

Analytics developed from automated measures of size and structural quality contributed to productivity and cycle-time gains during Fannie Mae's agile–DevOps transformation. A significant portion of the initial gain resulted from earlier detection of structural flaws and analysis of injection patterns, reducing corrective-maintenance effort. Aggregating these analytics allowed executive management to maintain visibility into the transformation's progress and empirically justify the investment in agile–DevOps practices.

Acknowledgments

Special thanks to Alan DeFelice for the visuals used in this article.

References

1. T. Menzies and T. Zimmerman, "Software Analytics: So What?," *IEEE Software*, vol. 30, no. 4, 2013, pp. 31–37.
2. D. Zhang et al., "Software Analytics in Practice," *IEEE Software*, vol. 30, no. 5, 2013, pp. 30–37.
3. N. Mellegard et al., "Impact of Introducing Domain-Specific Modelling in Software Maintenance: An Industrial Case Study," *IEEE Trans. Software Eng.*, vol. 42, no. 3, 2016, pp. 248–263.
4. G. Kim et al., *The DevOps Handbook*, IT Revolution Press, 2016.
5. B. Curtis, J. Sappidi, and A. Szynkarski, "Estimating the

Principal of an Application's Technical Debt," *IEEE Software*, vol. 29, no. 6, 2012, pp. 34–42.
6. *Automated Function Points (AFP), Version 1*, Object Management Group, 2014; www.omg.org/spec/AFP.
7. *Function Points Counting Practices Manual, Release 4.3.*, Int'l Function Point Users Group, 2009.
8. A. Elbanna and S. Sarkar, "The Risks of Agile Software Development: Learning from Developers," *IEEE Software*, vol. 33, no. 5, 2016, pp. 72–79.
9. G.A. Campbell and P.P. Papapetrou, *SonarCube in Action*, Manning, 2014.

ABOUT THE AUTHORS

BARRY SNYDER is a product manager for the DevOps release pipeline, code quality, and machine learning in Development Services at Fannie Mae and a key contributor to the DevOps transformation. His research interests are DevOps, enterprise transformation, and machine learning. Snyder received a master's in public administration in IT systems from Pennsylvania State University. Contact him at barry_t_snyder@fanniemae.com.

BILL CURTIS is a senior vice president and chief scientist at CAST and the executive director of the Consortium for IT Software Quality (CISQ). He leads research on software measurement, quality, and pathology, as well as maturity models and process improvement. He leads development of international software quality standards through CISQ. Curtis received a PhD in organizational psychology and statistics from Texas Christian University. He's an IEEE Fellow. Contact him at curtis@acm.org.