Quest

DevOps Comes to Database Developers and DBAs — It's About Time

Written by Daniel Norwood, Director of Product Marketing, Quest, and Robert Reeves, CTO, Datical

INTRODUCTION

Application teams have tools for managing their development lifecycle — from source code control and automated testing to continuous integration and automated deployment. But as a database developer or DBA, you've lacked the proper tooling to help you keep pace with the rest of your organization.

You face increasing pressure to deliver database changes faster, while you're also responsible for ensuring that databases are always available and applications run at peak performance. The conventional wisdom dictates that you have to choose between speed and quality, that you can't go faster and reduce risk at the same time.

Wrong.

Now your organization can apply automation to your database development lifecycle, breaking down silos between development and operations teams and bringing you into the DevOps fold. With the right tools, database developers and DBAs can finally keep up with the increasing pace of application releases while protecting precious business data.

This white paper will explore the importance of bringing agile and DevOps to database development and the obstacles organizations face along the way. It describes what to look for when selecting tools to bring the database into the world of DevOps. Finally, it demonstrates how the right tools can bring automation to database development and the value of DevOps to the entire organization.

67 percent of survey respondents said that they were purely agile or leaning toward it.

BACKGROUND: AGILE AND DEVOPS

Agile development is at the core of DevOps and the faster pace of application releases.

Agile solves the development bottleneck ...

In a Tech Beacon survey called "Is agile the new norm?", 67 percent of respondents said that they were purely agile or leaning toward it. Companies turn to agile to lower or eliminate the barriers between the business and development. They find benefits of agile development in all of their most important metrics: enhanced collaboration, higher software quality, greater customer satisfaction and lower cost of development.

With agile, new features and bug fixes get to market much faster and are on point with customer requirements. Gone are the days of spending a year and a half at the drawing board and coming back with a product; mobile app release schedules have set the expectation that users should wait no more than a couple of weeks or days for changes.

... and DevOps solves the release bottleneck ...

As development teams became more agile, it naturally led to the use of more software tools for process automation and collaboration. Eventually, the collaboration reached past the point of deployment and into operations. As depicted in Figure 1, the development and operations teams have grown to work together to manage code from design through operations and back in a truly collaborative cycle.

As collaboration improved and more processes were automated, the value delivered to the organization increased as well. The Puppet Labs State of DevOps Report shows that the value goes far beyond speed of delivery. Compared to companies that have not adopted DevOps, high-performing IT organizations enjoy 60 times fewer failures and recover from those failures 168 times faster than their lower-performing peers. They also deploy 30 times more frequently with 200 times shorter ramp-up.

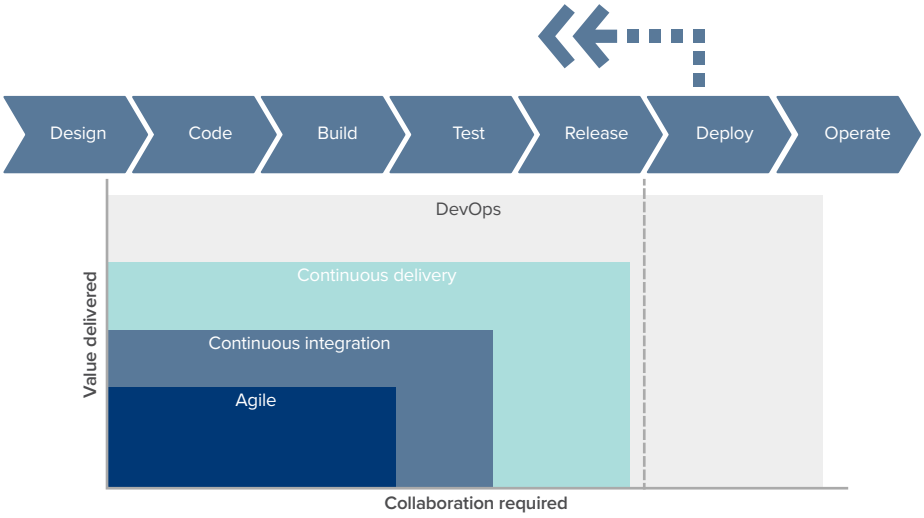

Figure 1: Evolution from agile to DevOps

Quest

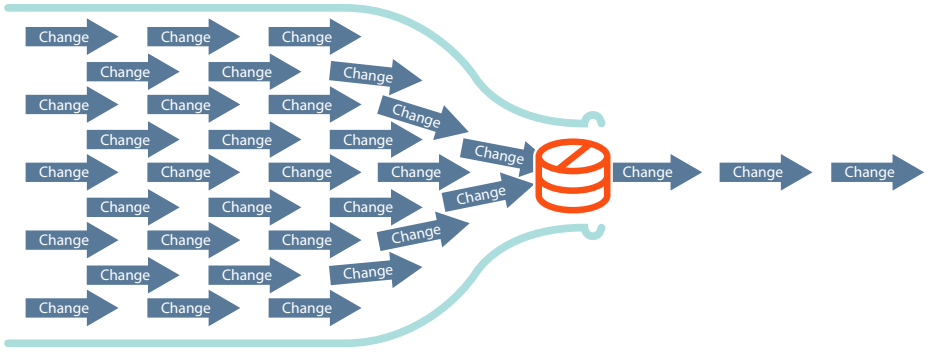

Figure 2: The process of database change has become the bottleneck in an otherwise agile process

... so what about the database development bottleneck?

Agile and DevOps have brought about great progress in application development through the use of automation tools and software. However, those tools are not a good fit for the work of database developers and DBAs, and the lack of proper tools leaves them out of the agile movement.

Organizations cannot enjoy all the benefits of true agile and DevOps if they do not include the database. Application releases may be ready as often as every few minutes, but as Figure 2 shows, if they depend on database updates that take place every month or quarter, then the organization as a whole misses the value of agile.

WHY IS DATABASE DEVELOPMENT SLOWER?

Given that database development needs to join the rest of the organization, what gets in the way of accelerating it?

The answer goes to the nature of databases and database development:

- For standard application development, making changes to the front-end application code is often more forgiving because it is possible to revert to a previous build if something goes wrong. There may be some service interruption, but the risk of losing data is usually low.
- With database development, it's necessary to preserve the state of the database and protect its data throughout code changes. This is accomplished with complex scripting based on the database's current state at that time.
- Whereas application developers have the option of reverting to a previous version, database developers have to jump through many more hoops to undo changes, even restoring the entire database in worst cases.

Application teams typically check in their code, and then a series of automated processes follows: build, test, review, stage, deploy. The changes propagate automatically through test environments and eventually into production as they pass automated validation. However, in cases where there are dependencies on database code changes, the rest of the organization must wait because processes at this level are still manual.

Going faster and lowering costs without increasing risk

The reasons for going agile are to reduce risk, lower cost and increase speed. These also apply to automating the database development cycle.

The dominant theme in database development has been the trade-off between speed and quality (risk). In order to avoid risk, database developers work methodically and carefully, looking through every script in an attempt to reduce risk. That does not lower cost or increase speed.

Nor does it make the business more responsive. If database changes are the bottleneck and slowing them down is seen as the best way to protect the database, then organizations that crave stability will never be completely agile.

Organizations cannot enjoy all the benefits of true agile and DevOps if they do not include the database.

If database changes are the bottleneck and slowing them down is seen as the best way to protect the database, then organizations that crave stability will never be completely agile.

DATABASE DEVELOPERS AND DBAS STUCK IN THE MIDDLE

What suffers when developers try to work faster?

- **Best practices** — All professional programmers know that there are certain practices to follow to write code properly. When time does not allow, those practices often go unobserved.
- **Proper testing** — In the rush to get code out, many development teams feel there is no time to create proper unit tests. As a result, testing at the developer level becomes limited to debugging code and ensuring that it is not faulty. That is necessary as a first step, but it is not enough. The loss of unit tests increases dependency on test teams later in the process, slowing the release process. Ironically, cutting this corner actually slows things more.
- **Code reviews** — Although they cannot afford it, DBAs spend a lot of time in code review meetings. They and their managers find themselves playing the role of code police, trying to uphold code quality one line at a time.

Even after navigating obstacles and trying to build up their team for faster development, most DBAs and database developers are able to trim release cycles down to two or four weeks at best. That kind of improvement is of little help when the application development team is comfortably releasing several times a day.

In short, the dilemma is to find the right automation tool that will speed up database development and lower cost without risking the integrity of the data.

AUTOMATION TOOLS FOR FASTER DATABASE DEVELOPMENT WITHOUT COMPROMISING QUALITY

The combination of Toad and Datical DB extends the benefits of DevOps automation to the database environment. It offers database developers and DBAs the opportunity to accelerate development while mitigating risk and inherently reinforcing best practices. Toad and Datical work together to create an agile pipeline for database changes at each step in the database development lifecycle (see Figure 3).

1. Track discrete changes to code (Toad)

Some developers simply rely on pulling the master version of source code from the database, but how do they know it wasn't changed without their knowledge? And what was changed before they got there? What happens if two developers need to change the same piece of code? Managing code artifacts in a version control system (VCS) is the best practice and foundation for speeding things up.

A VCS is a reliable means of ensuring that the right version of code is on its way into the build system and that the code object has full integrity. Version control also ensures that multiple developers in a live environment are not overwriting one another's changes. It is indispensable to application development, so Toad offers integration with systems like Git, Microsoft TFS, CVS, ClearCase, Perforce and SVN, right in the integrated development environment (IDE).

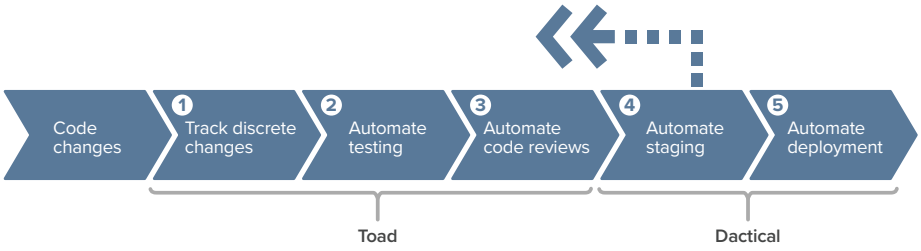

Figure 3: Toad and Datical — DevOps automation for database development

Quest

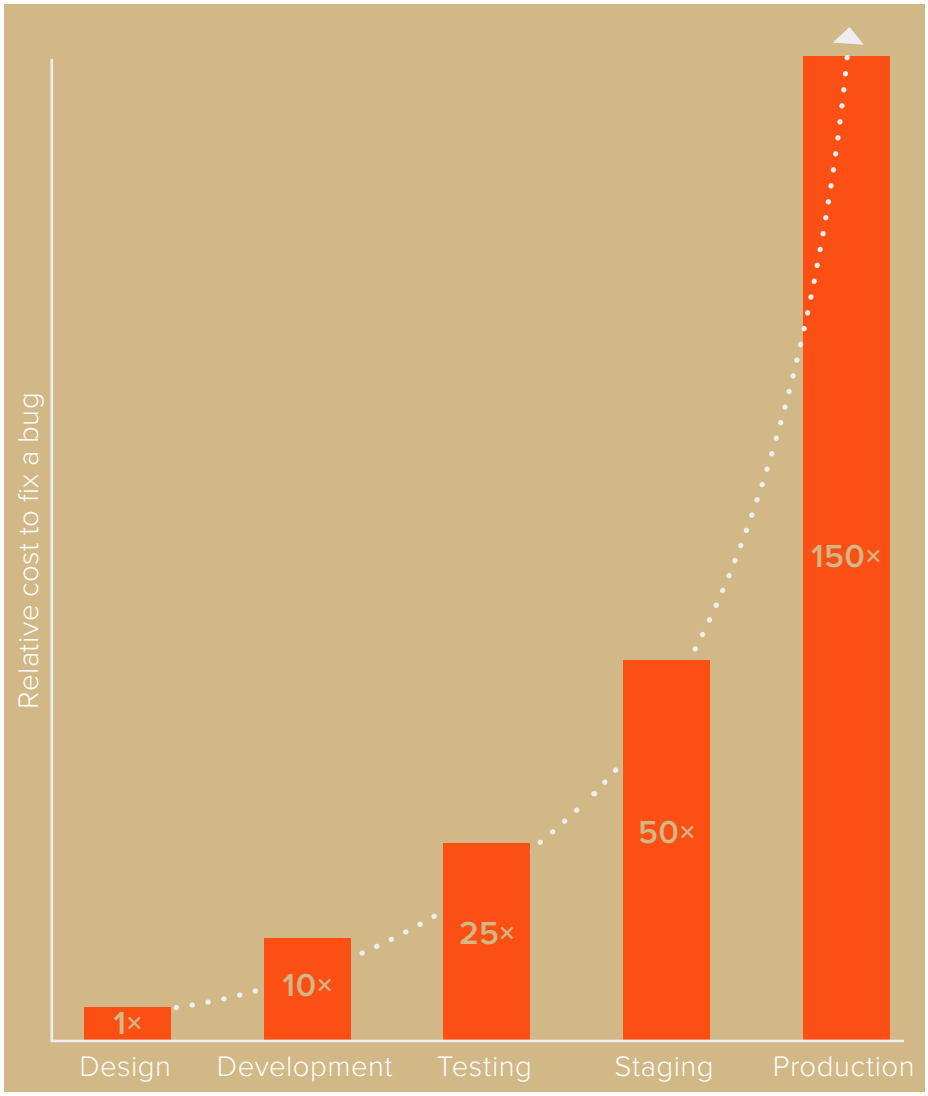

Figure 4: The Cost of Change curve

Development managers have the option to require PL/SQL unit test executions and minimum code review standards prior to code check-in, assuring quality without slowing things down.

2. Automate unit testing (Toad)

Once unit testing is automated, its value to DevOps becomes apparent.

In the rush to release, it's easy to inadvertently slow things down by passing bugs on and expecting subsequent processes to detect and fix them. Figure 4, the Cost of Change curve plotted by Barry Boehm, shows, in relative terms, the cost of finding and addressing bugs at different stages in the software lifecycle.

The cost ranges from jotting notes on the back of a napkin in the design stage to recalling a product, replacing it with an earlier version and losing precious time in the production stage.

Toad removes much of the pain at the development stage by simplifying the process of creating and running unit tests. Developers can debug as they normally would, save that information as a new unit test and accumulate the tests

In short, the dilemma is to find the right automation tool that will speed up database development and lower cost without risking the integrity of the data.

in a repository. Over time, they build up a regression testbed through which they can run all of their code changes. That reinforces the best practice of keeping code and test together.

From the Jenkins continuous integration (CI) build process, developers can call automated unit tests hosted on the Toad Intelligence Central server.

3. Automate code reviews (Toad)

Static code analysis and peer review are standard practices in application development, where automation tools for Java and other languages are common. But in the database world, the manual code review process is often a casualty of the rush to release.

To ensure that consistent standards of code quality are applied across an entire database project, Toad provides static code analysis from top to bottom against a set of more than 200 rules. The rules are built around the experience of the industry's top database professionals, and development teams can customize and add their own rules to enforce company standards. Toad applies the rules instantly in the editor — like a syntax check — so developers can see where they are violating code quality as they write. As an option, code must conform to the rules before check-in is permitted.

Automated code reviews reinforce quality naturally, without slowing down database development. The Toad Intelligence Central server offers managers a holistic view of code quality through consolidated web reports.

4. Automate staging (Datical)

After unit testing, code review and check-in, the next step traditionally is to hand off the scripts and code for staging. That requires manual review by the DBA.

Datical automates the enforcement of database compliance and governance requirements before committing the proposed change. It generates a summary of changes and an assessment of how they will affect the database. Datical applies a rules engine to enforce in-house practices, technical guidelines and regulatory standards. Here are some examples:

- Tables or functions will not be dropped in production.
- Every change should have a comment.
- Naming conventions must be upheld.
- Statements like DROP or TRUNCATE should not be used in stored procedures.
- Every change will be associated with a change management ticket.

When integrated with the build server, Datical can optionally fail the build if the rules are violated. Dealing with the problem then is less expensive than dealing with it in production, as shown in Figure 4.

Automating this step makes changing the database a part of the established, normal process of application development. The development team can move quickly but not at the cost of safety.

5. Automate deployment (Datical)

With the final stage of automating deployment, the database development process has the potential to move as quickly as application development does.

Datical is designed so that database developers can efficiently validate, deploy or roll back only those database changes necessary to support a specific capability or requirement as it is deployed. When there is a problem, they can segment large changes into the smallest executable steps, then immediately pinpoint the cause of pre-deployment warnings and failures. By tracking changes based on specific development activity, Datical reduces the possibility of human error and the time spent on database auditing and governance.

Toad provides static code analysis from top to bottom against a set of more than 200 rules from industry experts.

DATICAL

With Datical agile database automation, organizations can shorten the time it takes to bring applications to market while eliminating the security vulnerabilities, costly errors, data loss and downtime often associated with current database deployment methods.

CONCLUSION

Agile has addressed the development bottleneck and DevOps incorporates development and operations to create fully automated deployment pipelines. But the urge to go slowly and protect the database has led to a bottleneck that keeps organizations from realizing the overall value of agile and DevOps. When speed is seen as the enemy of quality, how can an organization avoid slowing things down in the name of protecting the database? How can teams speed up development without cutting corners or sacrificing best practices?

The right tools provide the missing link in an otherwise agile pipeline. A combination like Toad and Datical can bring the same degree of automation enjoyed by application developers to the database environment: automated change tracking, unit testing, static code analysis, staging and deployment.

Database development teams no longer need to choose among speed, quality and risk. They can release faster and respond to customers more quickly while maintaining quality.

FIND OUT MORE

For more details on how you can shorten development cycles without compromise, see the Quest e-book, Getting Agile with Database Development.

Learn more about Datical's approach to DevOps for the database at Datical.com.

> Managing code artifacts in a version control system is the best practice and foundation for speeding things up.

If you have any questions regarding your potential use of this material, contact:

Quest Software Inc.
Attn: LEGAL Dept
4 Polaris Way
Aliso Viejo, CA 92656

Refer to our Web site (www.quest.com) for regional and international office information.