Liquibase

WHITE PAPER

9 Recommendations for Mastering Database Compliance and Governance in Agile Environments

Mastering Database Compliance and Governance in Agile Environments

In today's hyper competitive application economy, more and more enterprise IT organizations are adopting Agile development methodologies in an effort to become more responsive to constantly changing business needs and to deliver high quality applications and services to market faster. For these organizations, Agile represents a fundamental change in how they plan, design, build and deliver applications and services to their customers. The change management process is no different, as noted by Gartner in a recent research report, which states, "Many I&O leaders' current change management processes were designed with regulatory compliance and risk mitigation as primary objectives. Speed was not stressed because it was not especially needed at the time."[1]

With the adoption of Agile, speed has now indeed become the focal point of the change management process. Enterprise IT organizations are able to deliver products and services faster, with greater frequency and with more tailoring towards customers' needs. But this newfound speed cannot come at the expense of safety, governance, or auditability in the database platform. For those database professionals supporting the enterprise's move to Agile, they now find themselves in the unenviable position of trying to support significant increases in release velocity while also maintaining the integrity of the enterprise's most precious asset—its data. Much like the move to Agile requires development teams to learn a set of new practices, workflows, and tools, so too will database teams need to adapt their processes and toolsets.

With the adoption of Agile, speed has now indeed become the focal point of the change management process.

In this white paper, we examine the differences between traditional software development approaches and Agile methodologies, how those differences impact compliance and governance in the change management process, and provide recommendations on how database teams can modernize their processes to support increasing release velocities while maintaining best practices in compliance reporting and data governance.

The Pros and Cons of Waterfall vs. Agile in Database Compliance and Governance

To fully understand the challenges that Agile methodologies, along with other, more recent philosophies like DevOps or Continuous Delivery pose to compliance and governance processes for database teams, it's necessary to understand and contrast the differences between Agile and more traditional approaches like Waterfall.

1 Gartner. *How to Evolve an Established I&O Change Management Process to Support Agile Development*. George Spafford & Ian Head, 09 April 2015

In Waterfall, the preparation for an annual or semi-annual release begins with an intense period of up-front design work. This phase, which can last from weeks to months depending on an organization's release cycle, ends when all Product Requirements Documents (PRDs), architecture diagrams, testing plans, etc. have been completed, revised, and approved by organizational stakeholders. This phase is then followed by the development phase. Once teams reach a point of completion in their development work they begin integrating features into the product, and validate their work to make sure that the requirements from the design phase have been satisfied and that all necessary documentation is in order. Then the testing effort begins, and the release candidate is promoted in one monolithic package through different environments on its way towards the final release state, when customers are able to realize the value of the team's work over the previous six to twelve months.

Because of its intense up-front design process and the strict adherence to the product documents that come out of it, Waterfall is excellent from a compliance and governance standpoint. Potential risks and security concerns for the release are identified during the design phase, and controls are put into place to mitigate those concerns and ensure the organization remains in compliance with the appropriate regulations. Another positive outcome of the design phase are the detailed product specifications that serve as a single source of truth, and help to keep development teams on the right track. Lastly, fewer production releases during the year means that each release is easier to manage and track, making it much easier to maintain an accurate audit trail.

But, in spite of these benefits to the compliance and governance processes, Waterfall is not as responsive to the needs of the business as Agile methodologies are. One reason for this is due to the amount of planning required in traditional approaches, which unfortunately leaves less time for execution of the plan itself, and elongates the release cycle.

Additionally, that intense up-front design effort leads to feature lock-in for development teams, inhibiting their ability to revise a particular feature set based on feedback from early users, or to respond to changes in the market that have occurred since the beginning of the current release cycle. Lastly, again due to the longer release cycles, customers are left waiting for new features for six to twelve months at a time, stagnating the customer experience and decreasing brand loyalty, which can potentially result in lost revenue for the company, particularly if a competitive product team is employing a more Agile approach.

Conversely, Agile development methodologies are much more responsive to the needs of the business. Working in short development iterations known as "sprints," Agile teams deliver product capabilities on a biweekly or monthly cadence, and focus on releasing a minimally viable set of features to their customers so they can gain user feedback quickly, which is then used to continually refine the product. Instead of a large up-front planning period, followed by another long period of development, Agile teams break that work up into more manageable chunks, where each sprint contains a mix of planning and execution. This approach allows for maximum flexibility in how the work is managed and executed, and greatly increases the responsiveness of IT to the needs of the business. For example, if conditions in the market change, or a competitor releases a new feature that is delighting users and stealing market share, Agile possesses the inherent flexibility to change the priorities of an upcoming sprint/s so that the team can focus on something new or different, which in turn allows the company to be as competitive as possible in the marketplace.

But, given the increase in flexibility and responsiveness that Agile provides the business, it carries some disadvantages to Waterfall in the areas of compliance and governance. For one, cycle times in Agile are measured in hours and days, and release targets are often changed due to the re-prioritization of feature sets. This significantly increases the difficulty in maintaining an accurate audit trail from the conception of an idea through its release. Additionally, because Agile emphasizes the collection of user feedback to refine the product, features can and often do go through several incarnations before finally solidifying, which again requires re-prioritization in sprint planning, and can render any up-front design or specification work obsolete. Without the original product documentation, it can be difficult to ensure that a particular feature contains the traceability or addresses security concerns in the way it was originally envisioned. Lastly, the management overhead associated with compliance and reporting makes it difficult to accommodate more frequent, but smaller, releases. When the work begins to pile up in an Agile environment, compliance and governance are easy items to skip in an effort to get the release out on schedule, which would be fine except for the fact that once one Agile sprint is complete, another immediately begins, creating a kind of organizational debt around maintaining an accurate audit trail.

So, in the end, it appears that making a choice between a Waterfall or Agile methodology means choosing between either predictable traceability, as you have in Waterfall, or speedy efficiency, which you get with Agile. In either case, the business is exposed to the specific vulnerabilities presented by whichever methodology is chosen. If the enterprise IT organization chooses a Waterfall approach, it will lose its responsiveness to ever-changing market conditions, which could potentially lead to lost customers and revenue. On the other hand, if the business chooses Agile, it exposes the organization's vulnerability to potentially falling out of compliance with the regulations it is subject to.

But this needn't be the case for the business that chooses Agile. In much the same way that a development team will need to adapt to support the move to Agile, Infrastructure and Operations (I&O) teams can gain both speedy efficiency and predictable traceability with Agile, but the I&O organization must modernize and adapt its compliance and governance processes to complement the Agile process.

Regardless of an organizations development approach, it is important to note that database audits are essential and inevitable. These audits involve observing a database to be aware of the actions of all database users. Auditors often set up auditing to investigate suspicious or malicious activity and for security purposes; for example, to ensure that those without the permission to access certain information do not gain access to it. However, the amount of time and effort required to find, gather and analyze all the information needed to satisfy an Auditors requirement can be significant, to say the least. This effort can last days, if not weeks, and pull DBA resources away from the critical task of promptly reviewing and promoting database changes required by the business.

To that end, the next section provides recommendations on how database teams can go about modernizing their compliance and governance processes in support of Agile development.

9 Recommendations for Mastering Database Compliance and Governance in an Agile World

1. Data Must be Complete

In making sure that your data is complete enough to satisfy any database compliance audit, it is necessary to collect more information than simply, "Bob ran this SQL script on this date against this database." It's important to know and understand what the business need was that made the change necessary, and to be able to align the database change/s appropriately.

The business need addresses the question as to "why" the change was made, and can be included in the database audit trail by recording information from the ticket request system along with the set of database changes. Types of information that should be appended with the database change include, but are not limited to:

- **The ticket number of the application change request:** This aligns the database change to the application change it is supporting, and makes lookup easier later on when an audit request asks why a particular database change was made.

- **The author of the change:** Recording this information makes it easier to find a knowledgeable person to speak to should the need arise, whether that's due to an audit request or a technical issue that needs to be addressed.

- **The package or release of the application (the application version) that the database changes were shipped with:** Aligning a package of database changes to the version of the application being supported makes it much easier to categorize and organize those SQL scripts, making lookup easier for compliance audits. But it also provides a history of the evolution of that database that is tagged to the application it supports.

- **Other environments where the database change has already been executed:** Recording this information helps with dependency management and troubleshooting during the promotion process.

2. Collection Must be Automatic

Many of our customers today are already tracking the necessary information to form a complete audit trail and satisfy compliance audits, in the form of spreadsheets, wikis, or ticketing systems that are updated with information after each database deployment. But as release velocities have accelerated due to the adoption of Agile development or Continuous Delivery, they find it increasingly difficult to accurately maintain these manual tracking systems. What's worse, there is greater complexity in managing Agile releases as priorities are constantly in flux, making it even more difficult for database teams to maintain an accurate picture of what has been deployed where, when, and by whom.

On top of the difficulty in maintaining an accurate record of change is the fact that manual tracking systems are relegated, by their nature, to tracking only one project or one team's activities at a time, rendering it virtually impossible to gain an accurate picture of the state of database deployment across the enterprise. This lack of visibility across the enterprise, in turn, makes it difficult to set and enforce database standards, as well as preventing the kind of analysis required to improve efficiency across the board.

The answer to this conundrum is for database teams to embrace an automated system of data collection. Automation can track all of the information that might potentially be required to satisfy an audit at the point of a database deployment, as it occurs. An automated system for data collection also prevents any human error, the probability of which increases as the release cycle accelerates.

3. Storage Must be Centralized and Accessible

Again, with manual collection of database change data through spreadsheets or ticketing systems, the data is spread out across different projects and teams, and piecing together the audit trail can become difficult because the data is housed in different locations and in different formats. This, in turn, increases the amount of time it takes to gather the information to satisfy an audit requirement, increases the amount of management overhead in the process, and steals time away from actually executing the work.

4. Data Integrity Must be Protected

In any automated system, it is necessary to control who has access to ensure data integrity is kept intact.

5. Create a Standardized and Repeatable Deployment Process

Today, much of a DBA's time is spent running SQL scripts to update a particular environment. Depending on the state of that environment—or what else has been run—executing that SQL script can be an exercise in finding dependent scripts, running those first, updating this particular script, etc. In this situation, there's really not a single, repeatable way to update the database from environment to environment as long as those databases are out of sync and the database team does not have a deployment process in place which allows them to catch a database up from one version to another. When every database deployment is a unique snowflake, enforcing governance processes and standards becomes exceedingly difficult.

6. Automate the Enforcement of Governance Roles and Policies

The data governance process is rendered ineffective if the enforcement of roles and policies is accomplished after the fact. As an example, discovering four days after the fact that someone received a database grant or permission who wasn't supposed to still means that the organization was out of compliance for four days. Automation can significantly help the database team in this instance by quietly enforcing roles and policies behind the scenes and before any compliance infractions occur.

7. Customize Database Standards to Meet the Unique Requirements of Each Team

Every database team is going to process their work a little differently, and each database team will be subject to a different set of internal and external regulations. This is fine, and as it should be, but also means that when it comes to validating that a set of database standards have been met, the process, whether manual or automated, must have the ability to be customized to that particular database team's set of standards.

8. Consistently Apply Integrity Checks

It's critical that database standards and rules are applied consistently every time an update is performed in order to ensure the integrity of the governance process. Here again, an automated approach can greatly assist the database team, but even a manual effort will be satisfactory as long as the checks are applied consistently across the board. The only problem with a manual approach is that humans tend to make more mistakes the faster they try to go, and as the release cycle accelerates due to Agile adoption the probability of human error increases significantly.

9. Centralize Governance Data

Storing governance data in one, centralized location makes it easier to query the data and ensures better data normalization, which leads to easier and more accurate reporting on governance metrics. Easier and more accurate reporting, in turn, leads to better validation as to whether the governance process is effective or not.

Liquibase Enterprise was Built for Agile

Liquibase Enterprise was purpose-built for database teams supporting Agile development, Continuous Delivery, and DevOps patterns. The use cases and workflows align to Agile development workflows and patterns, and Liquibase's out-of-the-box integrations with build and deployment automation suites ensure the kind of velocity database teams need to manage, track, and execute database deployments in an Agile environment. Most importantly, Liquibase Enterprise enables this kind of speed for the database team without sacrificing the safety and control that DBAs need to ensure that the security and integrity of the data is protected. The guardrails that Liquibase Enterprise puts in place around the database also serve to automate the collection, storage, and organization of the data required to satisfy compliance and governance requirements. Liquibase accomplishes this through two critical capabilities—customizable, automated validation of SQL scripts, and the comprehensive, automatic collection of database deployment data.

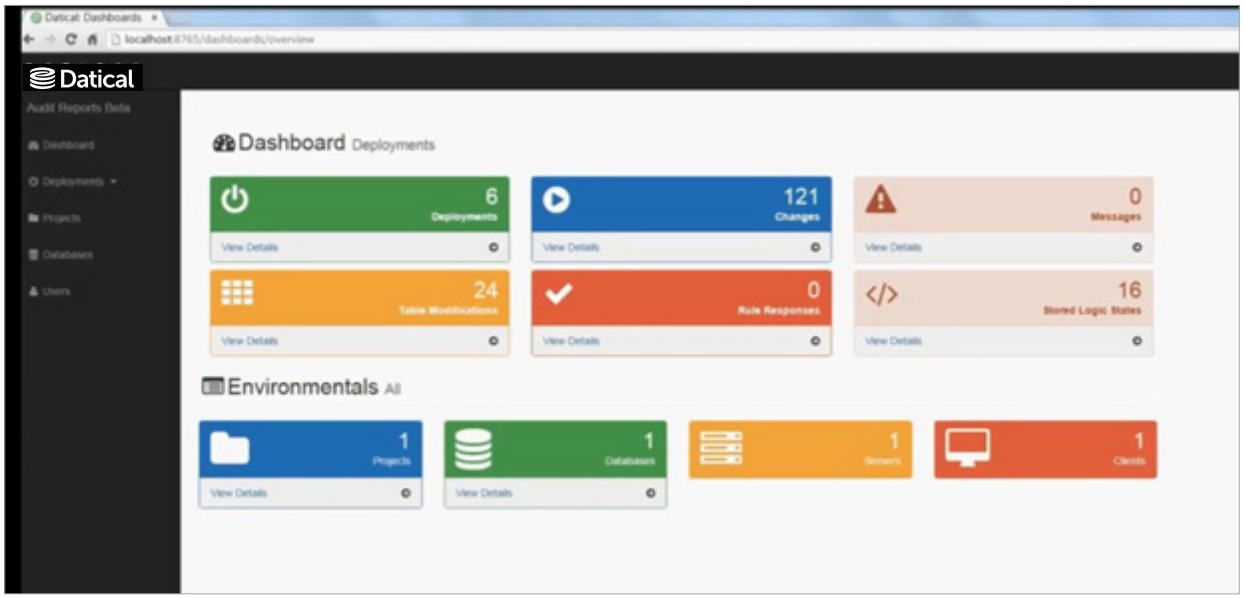

Figure 1: Liquibase Deployment Monitoring Console - Summary Dashboard View
Database deployment activity at-a-glance across all managed servers.

Capture and Enforce Business Rules

Liquibase's unique and customizable Rules Engine allows database teams to capture their best practices, corporate, and regulatory database standards directly into the tool to automate the process of SQL validation. When database changes are brought under Liquibase management, they become part of a data model that Liquibase maintains for each database. That data model then allows Liquibase to simulate the impact of proposed database changes against a target environment before a deployment is executed. Through static and dynamic SQL analysis, Liquibase first conducts a series of logical checks of the database code to make sure that the changes are technically sound and will work in the target environment. For example, if a SQL script includes an operation to add a column to a specific table, Liquibase automatically validates that the table does in fact exist in the target environment, and that there isn't already a column by the same name. The Rules Engine then expands on this capability by allowing database teams to write customized validation checks to ensure that proposed database changes conform to the team's unique combination of corporate, technical, and regulatory database standards. For example, if a database team wants to ensure that every database change is aligned to a specific application change request to facilitate compliance reporting, they can write a custom validation check that will be automatically, and consistently, applied to every proposed database change, alerting the team anytime a discrepancy is identified.

Automatically Collect Database Deployment Information

The second critical capability that Liquibase provides to database teams is the comprehensive and automatic collection of database deployment data. When a database is brought under Liquibase management, Liquibase creates

a change log that automatically captures and stores information about every database deployment. To support Agile workflows, the information captured in the change log goes beyond "who deployed what, where" to include information about the business need and application version being supported, the database and operating system environment, and performance data around the deployment itself such as whether or not an operation was successful, how long it took, etc., that can assist database teams in troubleshooting and performance tuning.

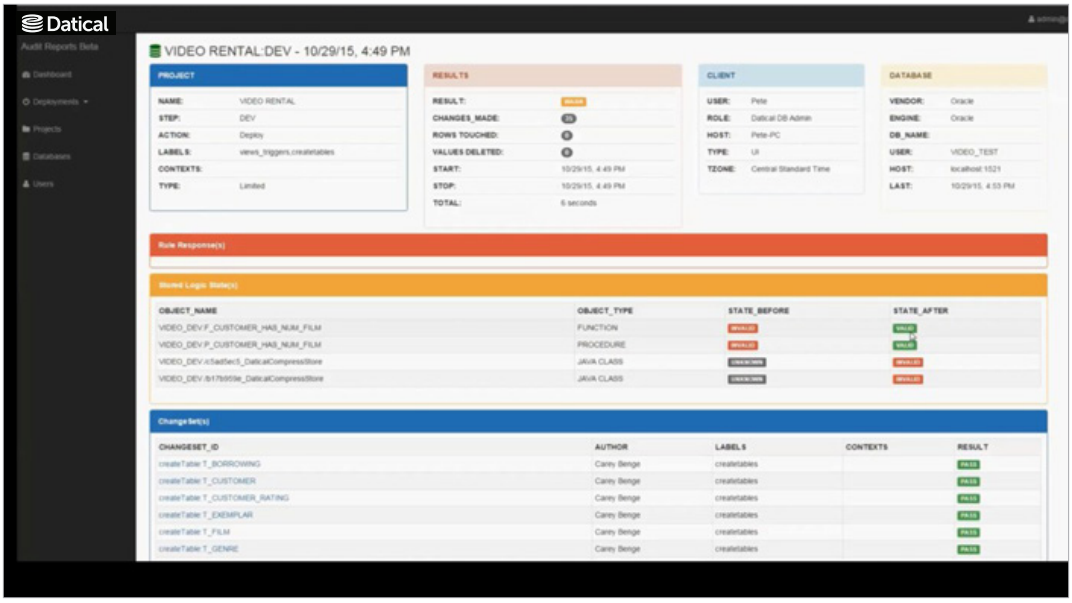

Figure 2: Liquibase Deployment Monitoring Console - Deployment View. Drill down to summary information for a single database deployment.

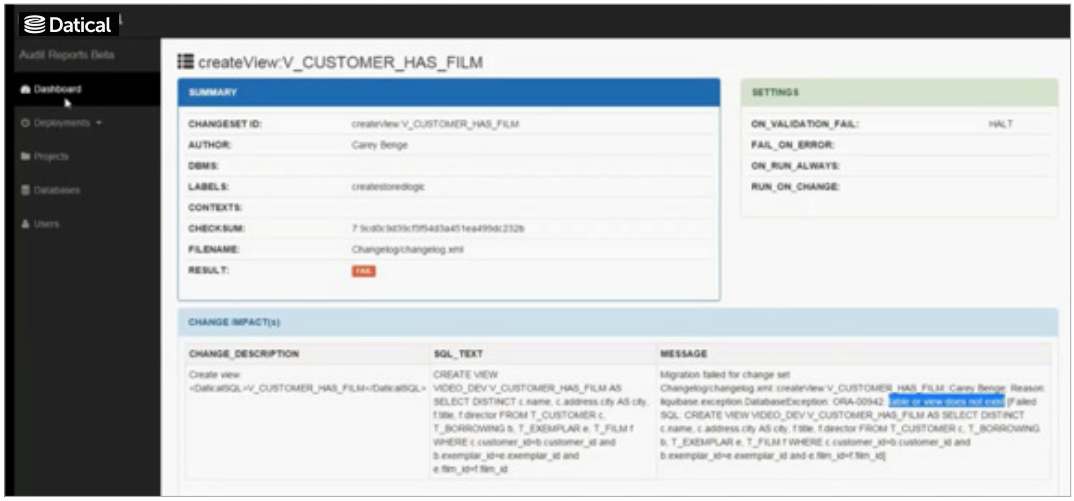

Figure 3: Liquibase Deployment Monitoring Console - Database Changeset View. Drill down to summary information for a single database deployment.

Centralize the Data that Will Be Required for Audits

Lastly, Liquibase centralizes the storage of deployment data across all managed database servers enterprise-wide with our newest capability, Liquibase Deployment Monitoring Console, to facilitate and automate compliance and governance processes. Tracking information at the project level, Deployment Monitoring Console maintains an audit trail for every managed database in an easy-to-query format that can either be accessed from the database console or fed into a business intelligence tool for analysis and aggregated reporting. Deployment Monitoring Console enables deep visibility into your database environments and provides instant status updates on database deployment activity enterprise-wide, like what changes have been run in which environments, the performance and status of those deployments, and information on any violations of the database standards that were identified via the Rules Engine.

To learn more about Liquibase and our Deployment Monitoring Console capability please go to:
https://www.liquibase.com/deployment-monitoring-console

About Liquibase

Founded in 2012, Liquibase's mission is to radically improve and simplify the application release process by automating database management. Liquibase solutions deliver the database automation capabilities technology executives need to get the most out of their Agile, DevOps and Continuous Delivery investments. With Liquibase 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.

Liquibase delivers results for some of the world's most admired companies including Bank of America, Wells Fargo, eBay, Fidelity Investments, American Express, GE, and McKesson.

www.liquibase.com | info@liquibase.com