Agilität & Datenbank: Eine Ehe mit Höhen und Tiefen

Markus Lohn
esentri AG
Ettlingen

Schlüsselworte

Agilität, Datenbanken, Software Entwicklung, Continuous Delivery, DevOps

Abstrakt

Im Rahmen von Projekten mit agilen Vorgehensweisen gibt es immer wieder Spannungen zwischen den Entwicklern und den "Datenbänklern". In diesem Zusammenhang sind viele Fragenstellungen zu betrachten, z. B. wie werden Änderungen an DB-Parametern, Tablespaces propagiert. Wer darf Tabellen und Spalten ändern, wie sehen die Verantwortlichkeiten aus, wie erfolgt die Verteilung auf eine Vielzahl von Umgebungen, welche Werkzeuge gibt es und wie sollte der Prozess aussehen, und .... und...

In diesem Vortrag werden diese verschiedenen Fragestellungen diskutiert und Lösungsansätze aus Projekten vorgestellt.

Einleitung

Software wird nicht zum Selbstzweck entwickelt. Softwareentwicklung hat das klare Ziel ein einsetzbares Softwaresystem zu produzieren. Die Software muss Anforderungen entsprechen, die aus Geschäftszielen abgeleitet wurden. Schlussendlich wird die Software für die Erreichung von Geschäftszielen erstellt. Geschäftsziele ändern sich stetig und eine Anpassung der Software wird daher notwendig. Aufgrund der immer schneller bereitzustellenden Funktionen und Services ist ein massives Umdenken in der Softwareentwicklung notwendig. Softwaresysteme müssen schneller, häufiger und in besserer Qualität zur Verfügung gestellt werden.

Aus diesem Grund muss sich die innerbetriebliche IT in der Zukunft anders aufstellen. Die gesamte IT in einem Unternehmen trägt die Verantwortung für die Bereitstellung von Infrastruktur und Anwendungen bzw. Services, um Fachabteilungen bestmöglich zu unterstützen.

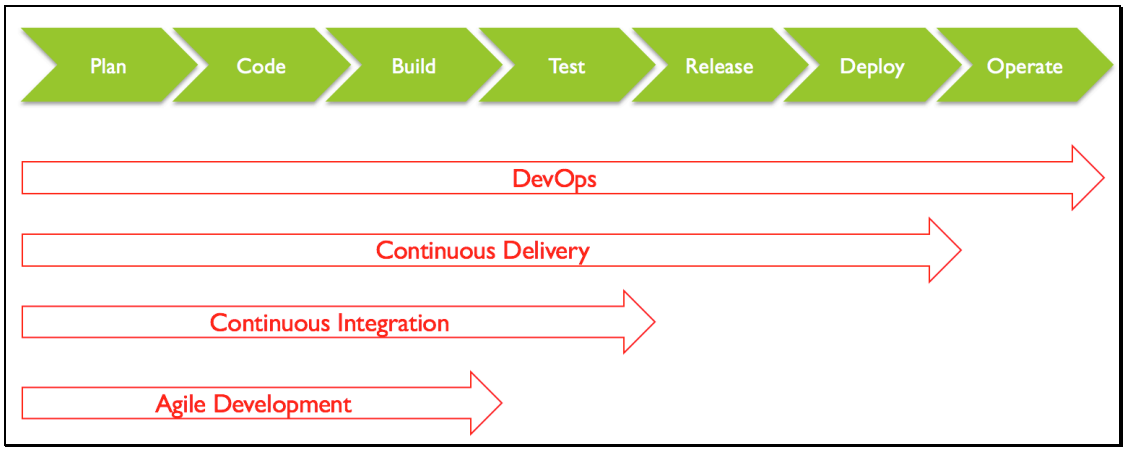

Abbildung 1: Lebenszyklus Softwareentwicklung

Die betriebliche IT, unabhängig von Abteilungen/Teams, muss dieses Ziel stetig verfolgen. Somit hat IT heute auch einen ganz anderen Stellenwert im Unternehmen. In den vergangenen Jahren wurden agile Methoden erfolgreich in der Entwicklung etabliert, um den bestehenden Anforderungen gerecht zu werden. Das alleine ist jedoch nicht ausreichend! Mit der DevOps-Initiative wurde versucht, die Barriere zwischen Entwicklung und Betrieb zu minimieren. In Bezug auf das Zusammenspiel zwischen Datenbankspezialisten und Entwicklung gibt es jedoch immer wieder Konfliktpotential.

In vielen Unternehmen sind Datenbankspezialisten immer noch in funktionalen Teams organisiert, die häufig dem Betrieb zugeordnet sind. Selten sind Datenbankspezialisten in der Entwicklung anzutreffen. Innerhalb des Datenbank-Teams gibt es oft eine weitere Unterscheidung in DBA und Anwendungs-DBA. Der DBA übernimmt dann allgemeine Administrationsaufgaben in Bezug auf die Datenbanken, ohne jedoch tiefes Wissen über die jeweiligen Anwendungen zu besitzen. Im Gegenzug übernimmt der Anwendungs-DBA Aufgaben, die Anwendungswissen benötigen. Idealerweise ist der Anwendungs-DBA auch „näher“ an der Entwicklung dran. Zwischen Entwicklung und Datenbankspezialisten existiert dann ein einfacher bis komplexer Prozess, der die Zusammenarbeit definiert und regelt. Dieser Prozess ermöglicht viele bis wenige Freiheitsgrade in Bezug auf Datenbankänderungen, die aus Sicht der Entwicklung benötigt werden. Das kann natürlich zu zeitlichen Verzögerung bei der Bereitstellung von Anwendungen führen. Datenbankspezialisten und Entwickler haben unterschiedliche Sichtweisen und Befindlichkeiten auf den Entwicklungs- und Bereitstellungsprozesses von Anwendungen. Somit gibt es viele Reibungspunkte und Konfliktpotentiale zwischen beiden Lagern.

Datenbanken sind auch heute noch ein wichtiger Bestandteil von Anwendungen. Vor allem relationale Datenbanken sind konzeptionell verstanden und am Markt etabliert. Die Wichtigkeit von Datenbanken ist auch auf Basis einer Lebenszyklus-Betrachtung von Daten und Anwendungen erkennbar. Daten und Anwendungen durchlaufen einen unterschiedlichen Lebenszyklus. Daten folgen typischerweise dem Schema: analysiert, verwendet, archiviert und gelöscht. Anwendungen starten mit der Phase Entwicklung und enden mit der Phase Abschaffung. Die Dauer des Lebenszyklus einer Anwendung beträgt ca. 10-15 Jahre. Bei Daten geht man von einem wesentlich längeren Zyklus aus (ca. 30 Jahre bei strukturierten Daten). Somit wird klar, dass sich Applikationen häufiger ändern, als die verarbeiteten Daten.

In diesem Vortrag wird das Thema Datenbank in agilen Projekten aus verschiedenen Blickwinkeln betrachtet und Denkanstöße für mögliche Verbesserungen gegeben.

Agilität & DevOps

Die Einführung von agilen Methoden in der Softwareentwicklung hat zum Ziel den Entwicklungsprozess flexibler und schlanker zu gestalten, die Effizienz zu steigern und am Ende auch Anwendungen schneller zur Verfügung zu stellen. Im agilen Manifest wurden 12 Prinzipien formuliert, die während der Entwicklung Beachtung finden sollten. Was bedeutet das nun für Datenbankspezialisten und Entwickler? Das agile Manifest legt einen starken Fokus auf Individuen, die ihre Fähigkeiten in einem Team einbringen. Das Team steht im engen Austausch mit der Fachabteilung, um sich Feedback über den aktuellen Stand einzufordern und neue Anforderungen entgegenzunehmen oder zu hinterfragen. Im Manifest ist keine funktionale Trennung des Entwickler-Teams vorgesehen. Vielmehr muss das Team aus Personen bestehen, die das erforderliche Wissen mitbringen, um die gestellten Aufgaben optimal erfüllen zu können. Warum nicht also auch Datenbankspezialisten in das Team integrieren? Interessanterweise existieren verschiedene agile Methoden mit unterschiedlichen Schwerpunkten. Je nach Methode werden unterschiedliche Projektrollen definiert. Als Beispiel wird in Feature Driven Development (FDD) die Trennung in Architekt, Chefprogrammierer und Entwickler vorgesehen.

Zusätzlich zur agilen Vorgehensweise in der Entwicklung hat sich mit DevOps ein weiterer Begriff in den letzten Jahren verfestigt. DevOps ist ein Kofferwort aus Development und Operations. Ziel der Initiative ist die Beseitigung der Bruchstellen zwischen Betrieb und Entwicklung. Neben den organisatorischen Aspekten geht es vor allem darum, gleiche Anreize, Prozesse und Werkzeuge aus der Softwareentwicklung im Betrieb ebenfalls anzuwenden. Z. B. können Konfigurationsskripte ebenfalls einer Versionskontrolle unterzogen werden und einen Build-Lebenszyklus durchlaufen. Warum sollten nicht auch Datenbankspezialisten SQL-Skripte in einer Sourcecodeverwaltung verwalten?

Architektur

Aus Sicht der Architektur besteht eine Anwendung aus verschiedenen Komponenten mit unterschiedlichen Verantwortlichkeiten. In der nachfolgenden Abbildung ist exemplarisch eine typische Java EE Architektur dargestellt. Aufgeteilt in 3 Blöcke unterschieden nach Client-, Middle- und Resource-Tier.

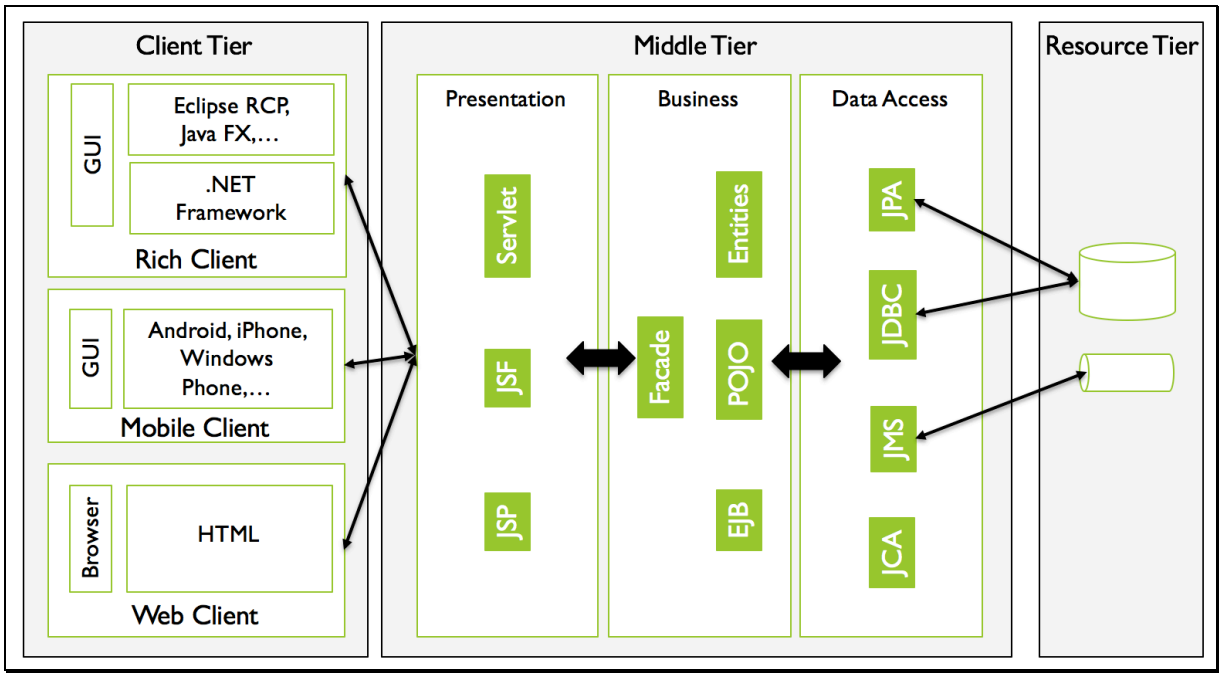

Abbildung 2: Java EE Architektur

Je nach Schicht kommen unterschiedliche Komponenten und Technologien zum Einsatz. Wird eine SOA-Architektur angestrebt kommen weitere Schichten und Komponenten hinzu, z. B. ein ESB, eine Prozess-Engine. Infrastruktur und Komponenten der Client-, Middle- und Teile der Resource Tier werden fast immer durch die Entwicklung betreut. Für die Datenbank existieren dedizierte Datenbankspezialisten. Eine Verbreitung des Wissens innerhalb des Entwicklungsteams bzw. andere Teamzusammenstellung können hilfreiche Maßnahmen sein.

Änderungen in der Datenbank bzw. Middleware im Vergleich

Die Datenbank wird immer als Fundament der Anwendung bezeichnet. Änderungen an der Datenbank sind angeblich immer mit Schwierigkeiten verbunden und erfordern immer den Einsatz von Datenbankspezialisten. Vergleicht man die unterschiedlichen Änderungsmöglichkeiten zwischen einer Datenbank- und Middleware-Anwendung sind viele Gemeinsamkeiten erkennbar. Der einzige und

wirklich große Unterschied sind Änderungen, die sich auf den aktiven Datenbestand in der jeweiligen Datenbank beziehen. Ein Datenverlust muss natürlich vermieden werden. Dieses Risiko besteht aber nicht auf allen Umgebungen im Entwicklungsprozess.

Änderung	Datenbank	Middleware / Applikation
Dateisystem	Tablespaces und Datenbank-Dateien	Domain-Verzeichnis
Memory	SGA	Java Heap Cache-Technology
Ausfallsicherheit	RAC	Cluster / Loadbalancer
Connection- und Threadmanagement	Datenbank-Listener	Java, Request Queue, Consumer Threads (Queues)
Sicherheit	Benutzer, Rollen und Rechte, Datenbank-Optionen	Benutzer, Rollen und Rechte, LDAP-Anbindung, Security Provider
Konfigurations-einstellungen	init.ora Parameter	JMS, JDBC, JTA, ... – Parameter
Objekte	Tabellen, Indexe, Sequences...	Queues, ...
Businesslogik	SQL, Trigger, PL/SQL-Packages	Java, Webservices, Prozesse

Beide Seiten erfordern gutes Verständnis und Wissen über die Auswirkungen von Änderungen. Die Herausforderungen sind für beide Seiten gleich: Richtige Konfigurationseinstellung, Objekte und Komponenten zum richtigen Zeitpunkt, mit der korrekten Version, auf der richtigen Umgebung bereitzustellen.

Entwicklungsprozess verbessern

Grundsätzlich ist die Datenbank natürlich ein sensibler Bestandteil einer Anwendung und bestimmte Änderungen müssen gut durchdacht werden. Allerdings können Änderungen an der Datenbank natürlich auch im Rahmen eines Projektes mit agiler Vorgehensweise problemlos organisiert werden. Jedoch ist es in einem solchen Szenario sinnvoll von bewährten Sichtweisen und Mustern ein stückweit abzuweichen. Hinsichtlich der Organisation ist es sinnvoll die Aufgaben zwischen Entwicklern und Datenbankspezialisten genau zu definieren und abzugrenzen. Idealerweise wird ein Datenbankspezialist einem Entwickler-Team zugeordnet. Das kann beispielsweise auch zeitlich begrenzt erfolgen und muss keine dauerhafte Lösung sein.

In einem aktuellen Projekt wurde beispielsweise definiert, dass Datenbankparameter, Tablespaces und Datenbankbenutzer immer durch die DBAs korrekt eingerichtet und zur Verfügung gestellt werden müssen. Die Entwicklung ist im Gegenzug für die korrekte Bereitstellung aller Datenbankobjekte innerhalb eines Schemas verantwortlich. Bereits während der Entwicklung wurden Datenbankspezialisten beim Design der Tabelle herangezogen. Während der Erprobung der Anwendung und Durchführung von Last- und Performance Tests wurden neue Teams aus Entwicklung und Betrieb gebildet. Diese Organisationsform unterstützte die lösungsorientierte Arbeitsweise sehr gut. Das gegenseitige Schuldzuweisen bei identifizierten Problemen entfiel komplett.

Ein ebenso wichtiger Punkt ist die Verwaltung von SQL-Skripten und Datenbankprogrammen (PL/SQL, Trigger). Es ist empfehlenswert diese Artefakte ebenfalls in der gleichen Sourcecodeverwaltung abzulegen, wie alle anderen Artefakte der Anwendung auch. Auf diese Ablage sollten Entwickler und Datenbankspezialisten gleichermaßen Zugriff erhalten. Nur dann ist sichergestellt, dass alle notwendigen Artefakte zueinander passen. Ferner können Änderungen jederzeit über die Historie nachvollzogen werden. Diese Vorgehensweise hat sich im Projekt bewährt. Vor allem in der DevOps-Bewegung wird dieser Ansatz auf den gesamten Betrieb ausgedehnt.

Um ständig in der Lage zu sein, Änderungen auf ein Produktionssystem auszurollen, ist Automatisierung zwingend erforderlich. Auch hinsichtlich der Verteilung von Datenbankänderungen ist das ein wichtiger Aspekt. In den letzten Jahren haben sich einige Werkzeuge am Markt etabliert, die unterschiedliche Aspekte von Datenbankänderungen berücksichtigen:

Puppet

Puppet ist ein Werkzeug zum Ausrollen von Konfigurationsänderungen und Installation von Software auf Servern im Netzwerk. Mit Puppet können Oracle Datenbanken installiert, gestartet und gestoppt werden. Insbesondere können Datenbankparameter geändert, Tablespaces und Benutzer verwaltet werden. Ein sehr interessanter Aspekt ist, dass Puppet die Konfiguration regelmäßig überprüft und Abweichungen ggf. wieder korrigiert.

Flyway

Flyway übernimmt das Management von Datenbank-Schematas über mehrere Umgebungen hinweg. Damit ist sichergestellt, dass das Datenbank-Schema immer passend zur Version der Anwendung zur Verfügung steht. Flyway kann auf verschiedene Art- und Weise in den Entwicklungsprozess eingebunden werden. Plugins für Maven, ANT werden ebenso zur Verfügung gestellt wie für Gradle. Damit kann es optimal im Rahmen einer Continuous Delivery Strategie einsetzt werden.

Flyway speichert die aktuellen Versionsinformationen für das Datenbank-Schema in einer Tabelle SCHEMA_VERSION und kann somit feststellen, welche Änderungen im Schema durchgeführt werden müssen. Die notwendigen Schema-Anpassungen können in SQL-Skripten oder Java formuliert werden. Flyway führt dann bei Bedarf alle notwendigen SQL-Skripte in der richtigen Reihenfolge gegen das Datenbank-Schema aus und aktualisiert die Tabelle SCHEMA_VERSION. Flyway kümmert sich somit um eine koordinierte Ausführung von SQL-Skripten. Die notwendigen SQL-Skripte können dann wieder durch das Entwicklungsteam in Verbindung mit Datenbankspezialisten erstellt werden.

Fazit

Die Datenbank ist in der heuten Systemarchitektur immer noch ein wichtiger Baustein. Daran wird auch der NoSQL-Hype nichts ändern. Jedoch dürfen Änderungen an der Datenbank nicht unterschätzt, aber auch nicht überbewertet werden. In der heutigen Schnelllebigkeit, muss die Softwareentwicklung schneller und flexibler werden. Datenbankänderungen sind also vorprogrammiert und sind im Entwicklungsprozess zu berücksichtigen. Für alle Komponenten einer Anwendung besteht das gleiche Problem: Alle Bestandteile der Anwendung mit der richtigen Version zum richtigen Zeitpunkt auf der richtigen Umgebung bereitstellen. Die agile Vorgehensweise unterstützt diesen Prozess und es gibt mittlerweile eine Reihe von Werkzeugen die den Prozess automatisieren helfen. Nicht zu unterschätzen ist jedoch wiedermal der Faktor Mensch. Eine agile Vorgehensweise ohne Veränderung der Kultur und ggf. auch der Organisation kann nicht zum Erfolg führen.

Kontaktadresse:

Markus Lohn
esentri AG

Pforzheimer Str. 132

D-76275 Ettlingen

Telefon: +49 (0) 7243 354900
Fax: +49 (0) 7243 3549099

E-Mail markus.lohn@esentri.com

Internet: www.esentri.com