MIDRANGE
MAGAZIN

11
2017

ITP VERLAG

IT-Konzepte und Wissen für POWER-Systeme im Unternehmensnetzwerk

Friedhelm Schnittker, Alos Solution

„Dokumente gehören den Vorgängen zugeordnet“

Schwerpunkt

...ufmännische ...wendungen

Technik

RCAC – Teil 5 Objektsignaturen

Friedhelm Schnittker, Vice President bei Alos Solution, im Interview auf Seite 24

Wie können Werkzeuge „Agile Development“ unterstützen?

Modernisierung der Software-Entwicklung

Die Anforderungen an die Verantwortlichen im Bereich Software-Entwicklung sind komplex und manchmal auch widersprüchlich hinsichtlich der Vereinbarkeit verschiedener Zielsetzungen. Kürzer werdende Releasezyklen zum Beispiel erfordern auch immer schnellere Reaktionszeiten der IT-Teams, um adäquat agieren zu können.

Dabei geht es zum einen um die schnelle Umsetzung neuer Anforderungen in geschäftskritischen Anwendungen, oft verbunden mit der Notwendigkeit einer Modernisierung der Anwendung zugrundeliegender Basistechnologie (Datenbank-Modernisierung SQL, Trigger, Stored Procedures, Web Services, Client Server Applikationen, Web Applikationen etc.). Zum anderen sollen aber auch die Kosten gesenkt werden, was massives Umdenken bezüglich der Strukturen und Abläufe erfordert, die sich typischerweise über viele Jahre in einem Unternehmen ausgebildet haben.

Um etablierte, aber nicht mehr zeitgemäße Prozesse abzulösen, sind innovative und effizientere Methoden im Bereich des gesamten Software-Entwicklungsprozesses notwendig. Es gilt, Software-Entwicklung als ganzheitlichen Ansatz zu verstehen, als einen Prozess, der sich vom Entwickler bis zum Endanwender durchzieht. Dabei rückt neben der Entwicklung verstärkt auch das Deployment für Testing und Produktion in den Fokus. Nur durch Zusammenarbeit aller involvierten Abteilungen kann das Ziel erreicht werden.

Die Verbesserung der Produktivität, die Automatisierung zahlreicher manueller, oft fehlerträchtiger Abläufe zur Steigerung der Effizienz sind dabei wichtige Aspekte. Das erfordert den Einsatz zusätzlicher Software, die zahlreiche bisher manuell ausgeführte Tätigkeiten übernimmt und zudem bisher nicht implementierte Anforderungen wie „Code Versioning“ abdeckt. Das Mittel der Wahl hierfür ist Agile Software Development, dessen Anfänge auf das „Agile Manifest“ aus dem Jahr 2001 zurückgehen und das auf zwölf Prinzipien beruht, die von 17 Autoren definiert wurden.[1]

„Agile Software Development“ kann als Oberbegriff für eine Reihe von Methoden und Praktiken verstanden werden, basierend auf den Prinzipien des Agilen Manifests. Dabei beschreibt Agile die Fähigkeit, so zu entwickeln, dass es gelingt, auch in unbestimmten und turbulenten Umgebungen reaktionsfähig zu bleiben. Lösungen werden hier durch die Zusammenarbeit zwischen sich selbst organisierenden, funktionsübergreifenden Teams entwickelt.[2]

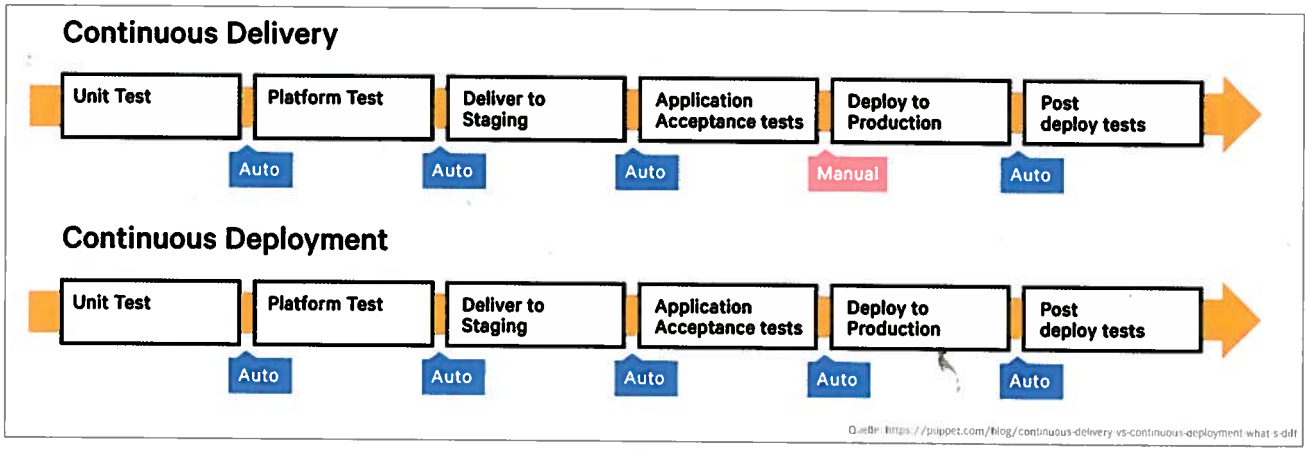
In der Realität des typischen „klassischen“ Projekts ist es in der Regel so, dass zu einem bestimmten Termin alle Entwickler ihre SW-Sourcecodes mitbringen und dann geschaut wird, wie man die Puzzleteile zusammenbringen kann. Die finalen Integrationstests laufen „unter den Vorzeichen der spannenden Frage“, ob das Level so in Produktion gehen kann oder sich mehr oder weniger gravierende Probleme zeigen? Die aus den Agile-Prinzipien resultierenden neuen Praktiken hingegen favorisieren eine andere Vorgehensweise, denen die Begriffe „Continuous Integration“, „Continuous Delivery“ und „Continuous Deployment“ zugrunde liegen.

[1] www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto

[2] www.agilealliance.org/agile101/ www.agilealliance.org/agile101/the-agile-manifesto

Wir erschließen bessere Wege, Software zu entwickeln, indem wir es selbst tun und anderen dabei helfen. Durch diese Tätigkeit haben wir Werte zu schätzen gelernt:

- Individuen und Interaktionen stehen über Prozessen und Werkzeugen
- Funktionierende Software steht über einer umfassenden Dokumentation
- Zusammenarbeit mit dem Kunden steht über der Vertragsverhandlung
- Reagieren auf Veränderung steht über dem Befolgen eines Plans



Quelle: https://puppet.com/blog/continuous-delivery-vs-continuous-deployment-what-s-diff

„Continuous Integration" bezeichnet eine Vorgehensweise, bei der alle Entwickler ihre Code-Modifikationen kontinuierlich in eine gemeinsame Umgebung „einchecken". Basierend auf diesem Quellcode-Level sollten täglich – wenn möglich automatisierte – Unit-Tests erfolgen, um sicherzustellen, dass die Veränderungen keine (ungewollten) Effekte in der Applikation erzeugen. Dazu kommen SW-Tools zum Einsatz, die ein kontrolliertes, nachvollziehbares Check-In und Check-Out ermöglichen. Im Kern zielt diese Methode darauf ab, die Feedback-Zeit für Entwickler signifikant zu verkürzen, um mögliche Probleme schnell erkennen und beheben zu können.

Continuous Delivery als geeignete Methodik

„Continuous Delivery" als Zielsetzung beschreibt eine Methode, bei der die verschiedenen Versions-Level jederzeit in Produktion gehen können. Damit wird „Continuous Integration" quasi vorausgesetzt und liefert die Basis der QA-getesteten Applikationsteile. Auch bei Continuous Delivery steht die Automatisierung der Prozesse im Mittelpunkt. Nur das Deployment, mit Schritten wie Kompilierung, Duplizierung von Sourcecode und Runtime-Objekten inklusive Roll-Out in die Produktion mit Rollback-Mechanismen erfolgt hierbei noch manuell. Mit dieser Implementierung kompensiert die IT-Abteilung einen zeitlich kritischen Faktor bei der Umsetzung geschäftlicher Entscheidungen.

„Continuous Deployment" ist die Weiterentwicklung von Continuous Delivery und erweitert die Prozesskette um die automatisierte Überstellung von SW-Change Requests in die Produktion. Somit ist die SW-Entwicklung jederzeit in der Lage, Änderungen in Produktion zu überstellen, zeitgleiche Integration vorausgesetzt.

„Continuous Integration" als weiteres Verfahren verkürzt vom Ansatz her nochmals die Check-in- und Test-Zyklen der Units. Erreicht wird damit, dass fehlerhafte Änderungen nur von begrenztem Impact sind und schnell behoben werden können.

Bei „DevOps" besteht die Zielsetzung primär darin, die Barrieren zwischen Development und operativem Management (Operation) zu beseitigen. Die bisherige Rollenteilung, dass Entwickler nur für die Erstellung der Software verantwortlich zeichnen und mit der Überführung in die Produktion nichts zu tun haben, soll hierbei überwunden werden. Entwicklung und Produktion sollen gemeinsam verantwortlich zusammenarbeiten.

Zusammenfassung

In der Praxis sind heute oft noch Kundensituationen anzutreffen, in denen sehr alte, nicht mehr zeitgemäße Vorgehensweisen in der SW-Entwicklung etabliert sind. Hier gilt es, eine Strategie der nächsten Schritte mit den richtigen Prioritäten zu entwickeln und umzusetzen. Oftmals ist hierfür zunächst die Etablierung von CM-Tools zur Analyse der Ist-Situation von Nöten. Denn bevor für die grundlegenden Prozesse keine strukturierte Vorgehensweisen für alle Entwickler definiert ist, wird es kaum gelingen, Software-Entwicklung moderner aufzustellen.

Enormes Potenzial im Markt für IBM i

An dieser Stelle sehen wir noch erhebliches Potential im IBM i-Markt. Gerade durch den Einsatz von Tools, die untereinander kommunizieren können, lassen sich relativ schnell Erfolge verzeichnen.

AXSOS als langjähriger Partner für den IBM i-Markt setzt hier auf starke Werkzeuge von Fresche Solutions und Remain Software, die durch Integration einen zusätzlichen Mehrwert bieten. Sprechen Sie uns an, wenn Sie mehr dazu erfahren möchten.

Die AXSOS AG mit Firmensitz in Stuttgart und weiteren Geschäftsstellen in Dettingen, Solingen und Ramallah kann für ihre Kunden auf ein umfassendes Produkt-Portfolio namhafter Hersteller wie zum Beispiel Computer Associates, Fresche Legacy oder Websydian zurückgreifen. **Michael Seifert** ■

→ **www.axsos.de**