HTWK Leipzig JUGSAXONYCAMP

Softwareforen Leipzig

Softwarequalität erhöhen durch DevOps

Leipzig, 31.03.2017

Jeremias Hackbeil

Nur wer schnell ist, überlebt im Markt.
Dafür braucht es neue Arbeitsstrukturen.

Computerwoche vom 03.06.2015

Foto: alphaspirit - Fotolia.com

HTWK Leipzig JUGSAXONYCAMP

Softwarequalität erhöhen durch DevOps

- **Worum geht es?**
  - Aufklärung / Verständnis von Begriffen
  - Softwarequalität
  - verschiedene Ausprägungen DevOps
  - Aufzeigen warum sich die Qualität durch DevOps erhöht
- **Worum geht es nicht?**
  - vollständige Definitionen der Begriffe
  - Einführung in Docker, Vagrant o.ä.
  - Wie führe ich DevOps ein?

Agenda

1	Begriffe
2	DevOps
3	Tools

Begriffe

Softwarequalität

- Gesamtheit der Merkmale und Eigenschaften eines Softwareprodukts, die die Erfordernisse erfüllen (nach Wikipedia)
- besteht aus Funktionalität, Zuverlässigkeit, Benutzbarkeit, Änderbarkeit, Übertragbarkeit und Effizienz (nach ISO 9126)

IT-Service-Management = IT-Betrieb

- Maßnahmen und Methoden zur Unterstützung der Geschäftsprozesse
  z.B. Bereitstellung von IT-Infrastruktur

Software-Entwicklung = Software-Engineering

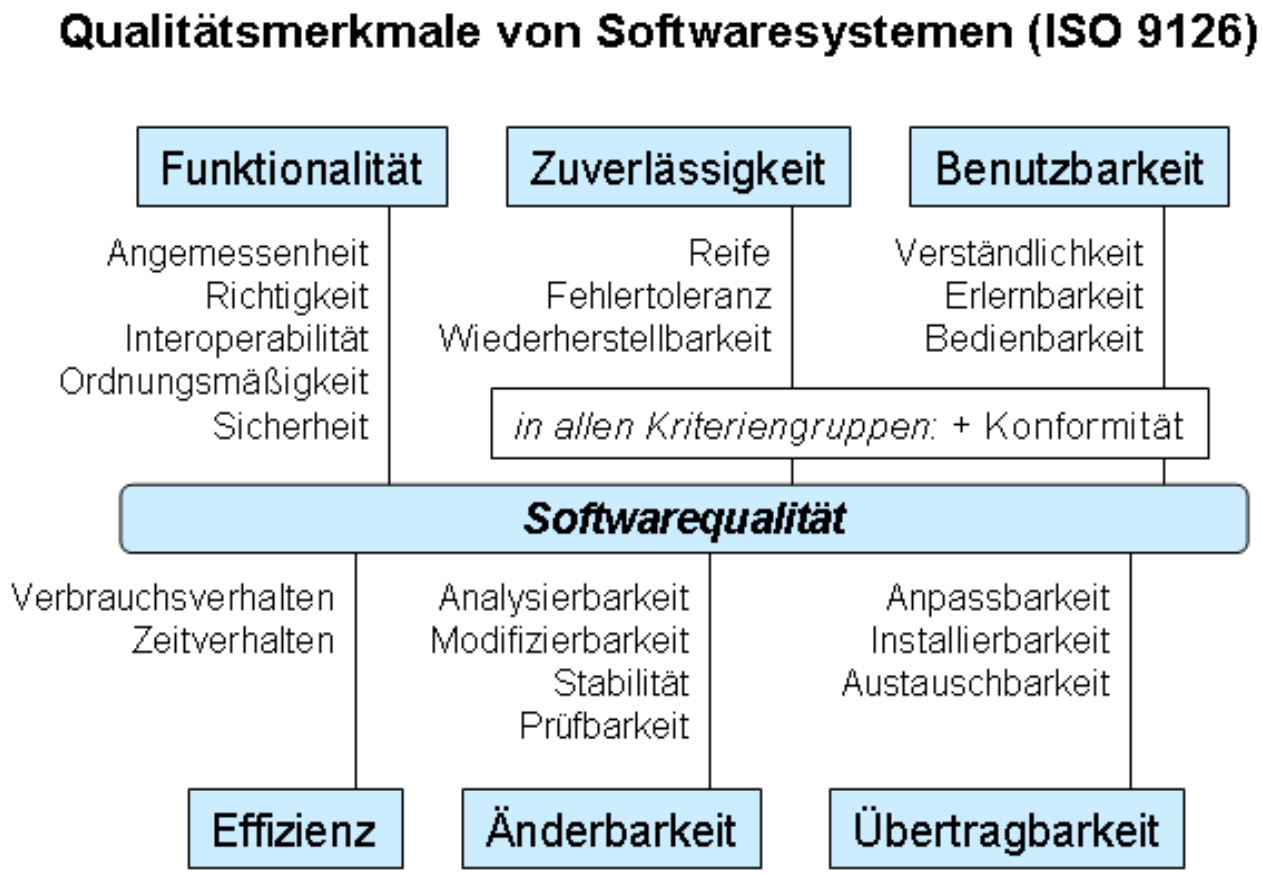
- Tätigkeiten zur Herstellung und Entwicklung von Software
- im weiteren Sinne alle Tätigkeiten, wie Projektmanagement, Qualitätsmanagement, Anforderungserhebung, Konzeption, Implementierung, Testen, Inbetriebnahme und Wartung
- in engeren Sinne nur Implementierung

DevOps

- kommt von ***Dev****elopment* und *IT* ***Op****erations*
- bezeichnet die Zusammenarbeit der beiden Bereiche

Begriffe

Softwarequalität nach ISO 9126



viele Kriterien und hohe Komplexität

Qualität in verschiedenen Bereichen

CleanCode

- variiere
- tue das Nötige
- isoliere Aspekte
- bilde wenig Abhängigkeiten
- halte dich an Angaben / Versprechen

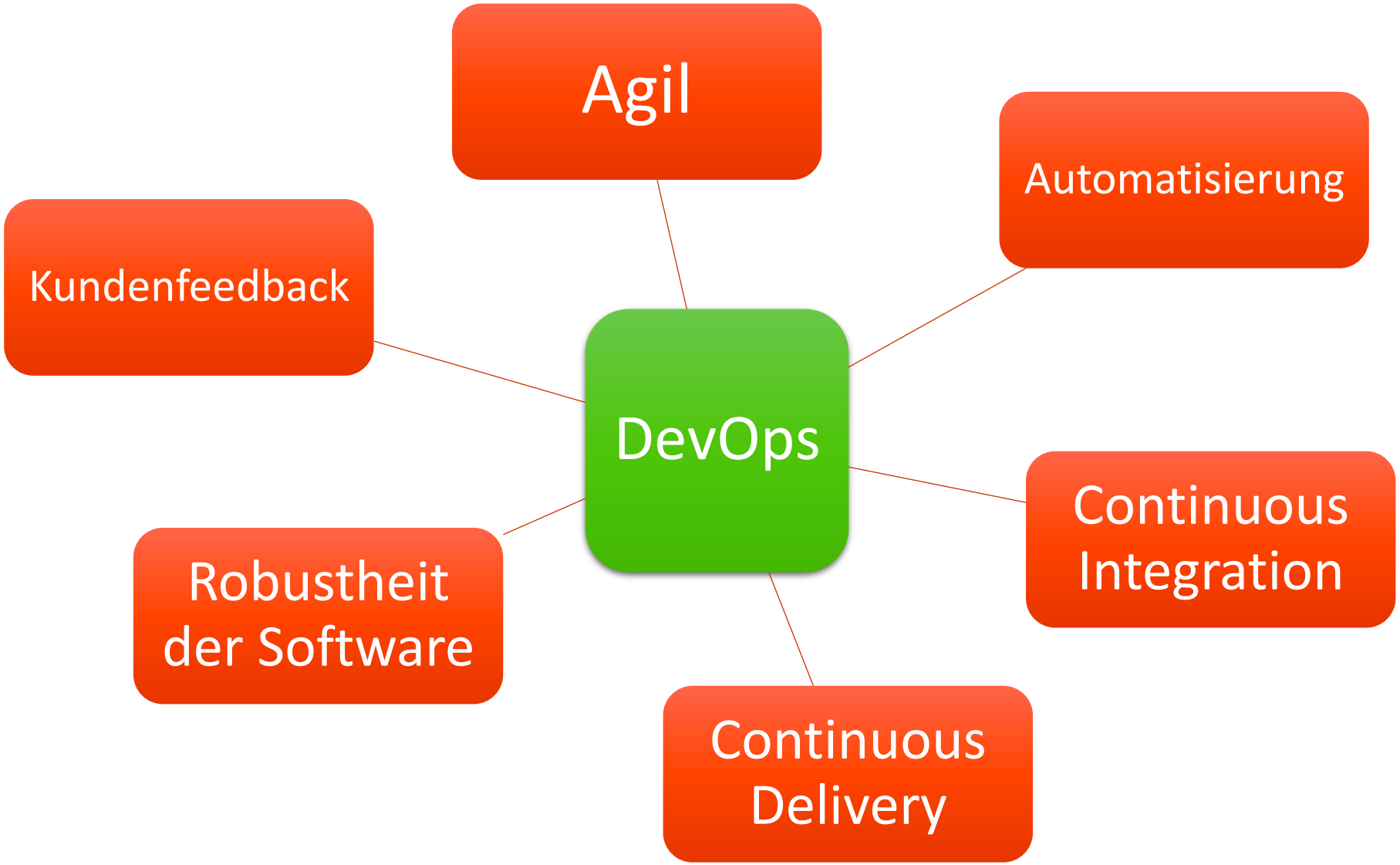
Architektur

- bekannte vs. neue Architektur
- Bewertung der Architektur
- Funktionale Gliederung
- Modularität
- Monitoring

Testen

- UnitTests
- Integrationstests
- Systemtests
- Akzeptanztests
- Regressiontest

Begriffe



Kontexte (vereinfacht)

kleinere Projekte / kleinere Unternehmen

- Einführung ist einfacher, weniger Entscheidungshemmnisse
- Erfolgserlebnisse sind schnell spürbar

größere Projekte / größere Unternehmen

- Einführung ist schwieriger, weil mehr Blockaden vorhanden sind
- mehr Formalisierungen, Standards, Spezialisierungen vorhanden
  → mehr Vorgaben, weniger Freiheiten

Wege zum Einsatz

schnell und Schritt für Schritt

- Pro
  - es kann gleich losgehen
  - Fehlentscheidungen werden schneller erkannt
  - an den Bedürfnissen ausgerichtet
- Contra
  - wirkt weniger Professionell
  - läuft evtl. „nebenbei“

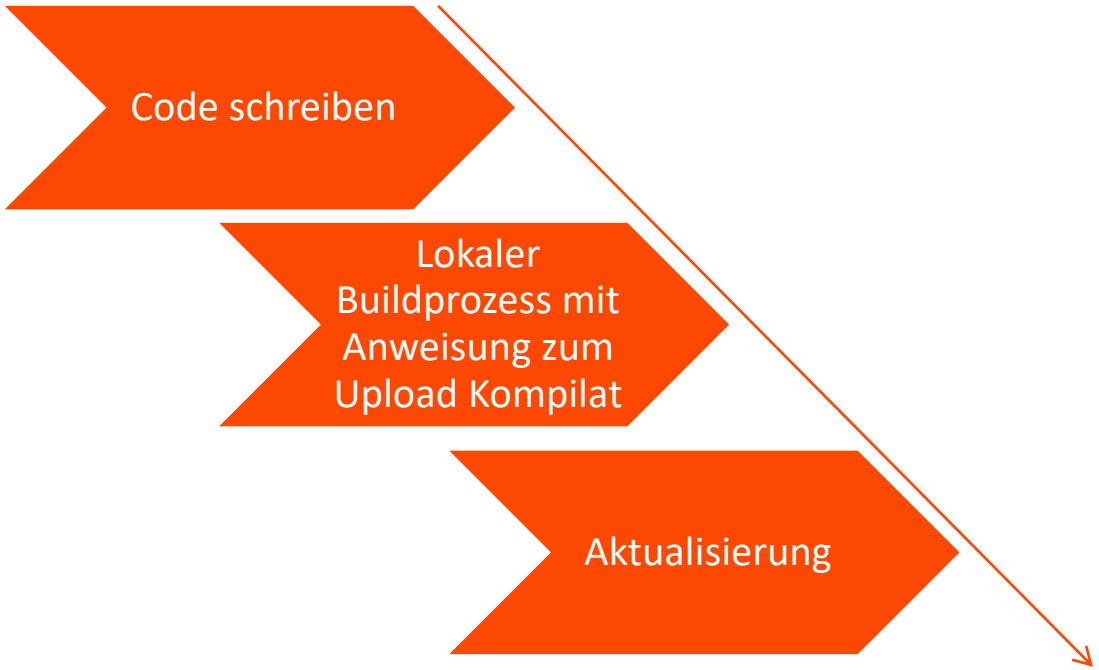
alles auf Einmal

- Pro
  - Konzentration liegt auf einem Thema
  - Alle Beteiligten können sich darauf konzentrieren
- Contra
  - lange Vorbereitung
  - alles theoretisch durchdenken

HTWK Leipzig JUGSAXONYCAMP

einfacher geht's nicht

Die One-Man-Show



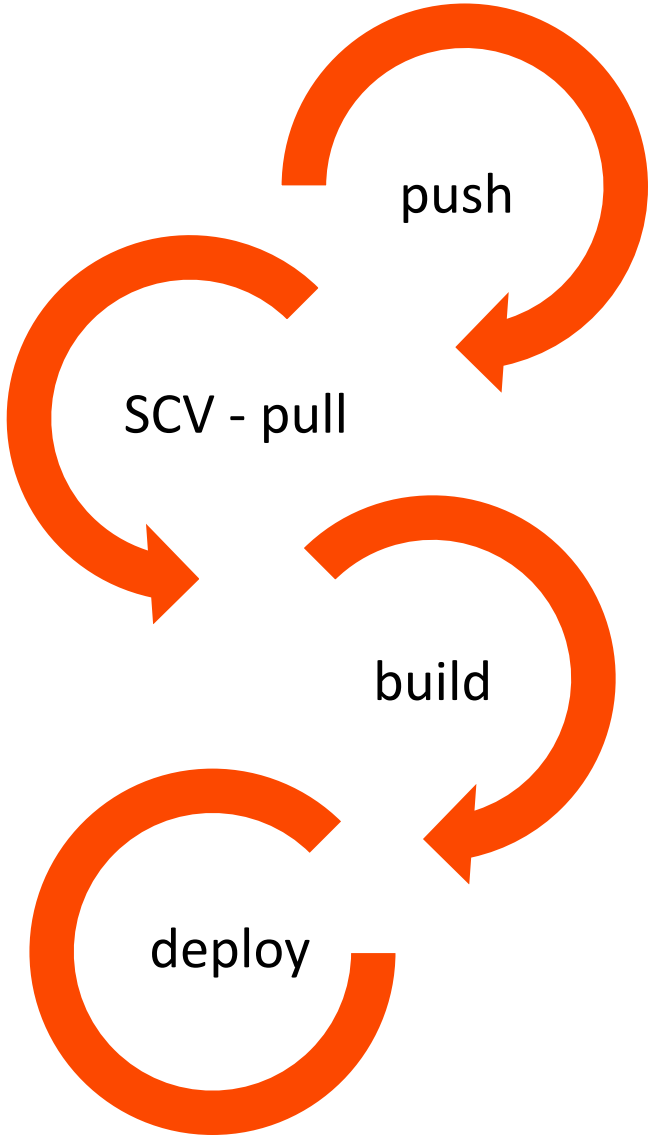
Maven

```
<plugin>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>exec-maven-plugin</artifactId>
    <version>1.1</version>
    <executions>
        <execution>
            <phase>pre-integration-test</phase>
            <goals>
                <goal>exec</goal>
            </goals>
        </execution>
    </executions>
    <configuration>
        <executable>scripts/start-agent.bat</executable>
    </configuration>
</plugin>
```

ant

```
<target name="-post-dist">
    <exec dir="d:/DEV/Projekte/Planning/" executable="C:\Program Files\Git\git-bash.exe">
        <arg value="rollout.sh"/>
    </exec>
    <echo>executed task: -post-dist</echo>
</target>
```

Nah am Entwickler



Entwickler: IDE

Übertragen des Code zum Buildserver

Buildserver baut Code, analysiert (und gibt Feedback)

Buildserver veröffentlicht auf Zielserver

Gesamtprozess

Feedback / Projekt-management

Atlassian Bitbucket JIRA Confluence

Entwicklung / Test

Deployment

sonarqube

eclipse IntelliJIDEA NetBeans

Qualität durch DevOps

- Hoher Grad an Standardisierung
- Automatisierung mit Abbruch bei Problemen (Tests, Quellcodeanalyse)
- Toolunterstützung
- häufige Releases möglich
  (nicht nur 2x pro Jahr)
- Strukturierte Aufgabenabarbeitung
  (je nach Entwicklungsmodell, zB Wasserfall, Kanban, Scrum, …)

Tools

Toolchain = Sammlung an Tools, die mit- und nacheinander arbeiten

- IDE (Eclispe, Netbeans, IntelliJ,…)
- Sourcecodeverwaltung (z.B. git, Bitbucket)
- Buildserver (z.B. Jenkins, Hudson, Bamboo)
- Quellcode-Analyse (z.B. SonarQube)
- Build-Tool (z.B. maven)
- Dokumentation (z.B. Confluence, Wiki)
- Ticketsystem (z.B. Jira, redmine, mantis)
- Virtualisierung des OS (z.B. Vagrant)
- Virtualisierung von Containern auf dem OS (z.B. Docker)

Ende

Softwareforen Leipzig GmbH

Hainstraße 16, 04109 Leipzig | **I** www.softwareforen.de | **E** info@softwareforen.de