Agile Software-Entwicklung: moderne Softwareentwicklung

Probleme treten auf, wenn z.B. in der Anforderungsanalyse ein Fehler gemacht wurde, so fällt dieser erst bei Bereitstellung auf und hat bis dahin viele Kosten produziert. Ein weiteres Problem ist, dass Softwareprojekte meist sehr großen Umfang haben und zu Beginn noch gar nicht klar ist, welche Anforderungen genau existieren und im Laufe des Projektes sich diese auch ändern können.
Dafür werden in den Phasen Anforderung, Architektur und Entwicklung **sogenannte agile Entwicklungsprozesse wie Scrum oder Kanban** genutzt. Dort werden nur kleine Mengen wichtiger Anforderungen ermittelt und anschließend umgesetzt. Das erzeugte Softwareprodukt wird dem **Kunden gezeigt und auf Basis dieses Feedbacks** werden dann die Anforderungen für die nächste Iteration gesammelt. Diese Phase hat nur eine Dauer von wenigen Wochen. Dadurch kann man mehr dem Wünschen des Kunden folgen. Das ist unter „agil" gemeint.

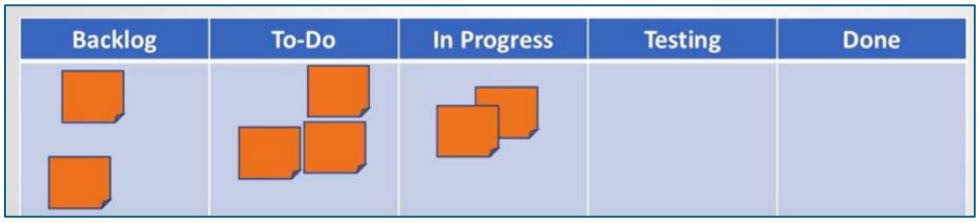
Damit diese Phase schnell abgeschlossen werden kann, wird das Testen, die Bereitstellung und der Betrieb **sehr gut automatisiert**. Dadurch wird hier viel zeit gespart und der Kunde kann schnell das Feedback danach abgeben, was wieder für eine Änderung wichtig ist. Diese Automatisierung hat eine hochgradige Technologie nötig, so dass alle Schritte schnell und damit sehr oft ausgeführt werden können. Diese schnelle Durchlaufzeit pro Feature ist sehr wichtig! Siehe später den Bereich „DevOps".

Scrum:
ist ein agiles Vorgehensmodell
übersetzt „Gedränge" beim Rugby. Alle Scrum-Mitglieder treffen sich täglich, um sich gegenseitig über den Stand der Dinge zu informieren und abzustimmen.

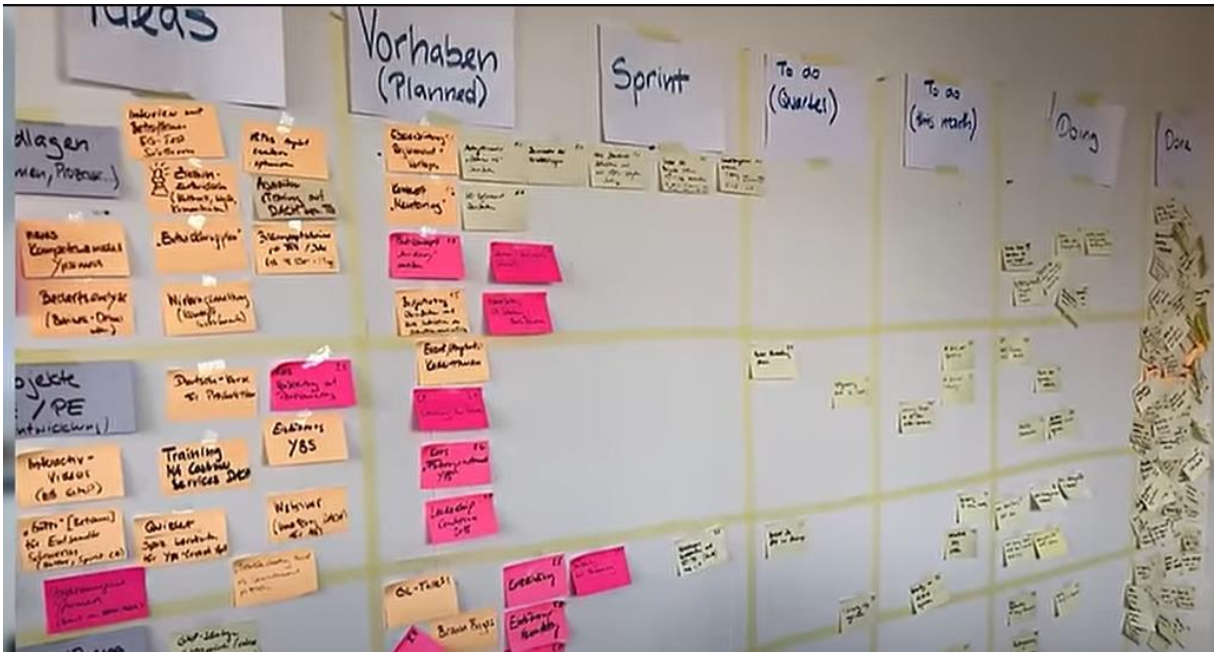
- Man arbeitet innerhalb von „sprints" (Zeitspanne von 2-4 Wochen)
- Ergebnis: fertiges Inkrement der Software
- Bestandteile:
    - unterschiedliche Rollen: product owner (gibt die Anforderungen vor und hat die Idee), scrum master, Entwicklungsteam (Backend-, Fronend-Spezialisten, Datenbank-Entwickler...)
    - Ereignisse: sprint-planning (Planung der Arbeit mit dem ganzen Entwicklungs-Team), daily-scrum, sprint-review, Retrospektive
    - Artefakte: product backlog, sprint backlog (Hier wird vom kompletten Team festgehalten, wie viel aus dem product backlog im kommenden Sprint realistischerweise umgesetzt werden kann), Produkt-Inkrement

Kanban:
ist ein agiles Vorgehensmodell – schneller Überblick
Ursprung: Toyota-Produktionssystem, optimaler Fluss durch die Fertigung von Autos
Prinzipien: visualisiere den Fluss von Arbeit – erstelle eine Kanban-Board: z.B. auf einer Tafel. Wichtig ist die Unterscheidung von links nach rechts der einzelnen Prozessschritte. Jeder Prozessschritt hat eine Spalte. Somit wird von links nach rechts der Workflow visualisiert.

Häufigste Mindesteinteilung ist: to do (noch nicht bearbeitet), in progress (wird gerade bearbeitet), fertig (finished)



Mit Post-It kann man dann die einzelnen Arbeitspakete von links nach rechts wandern lassen und weiß somit genau, wo sich etwas gerade befindet. Bis dann zum Schluss alle Schritte ganz rechts angekommen sind.



https://www.youtube.com/watch?v=4u4x4Zg5Zpg&list=PLNmsVeXQZj7qNMn6zimfu4JPUklG-4Uu4&index=75

Werden z.B. in einem Prozessschritt viele Pakete länger behalten, dann ist möglicherweise dort das Team zu klein und verursacht den Engpass.

Unterschied zu Scrum:
Bei Scrum sind die Sprints exakt mit Zeit vorgegeben, bei Kanban nicht.
Kanban hat keine festen Rollen wie Scrum.

Beispiele für Projektmanagement-Tools:

- Jira
- Trello
- GitHub
- Asana
- ClickUp

modernste Form in der agilen Software-Entwicklung

DevOps (Development Operations)

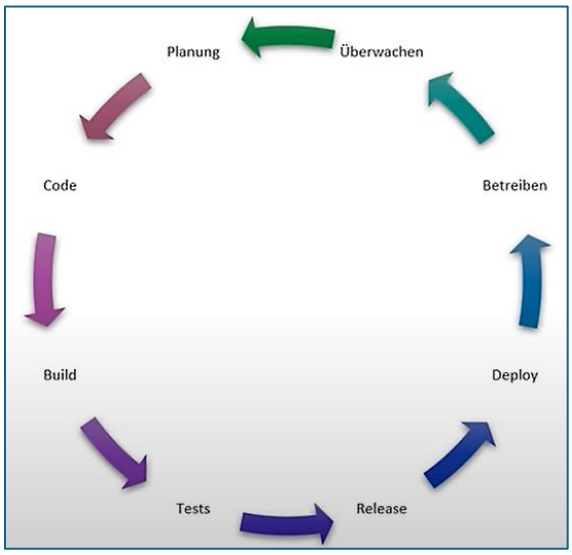
als Kombination von den Phasen der Entwicklung (Anforderung, Architektur, Entwicklung und Testen) mit denen des Betriebs (Bereitstellung und Betrieb). Das ermöglicht eine schnelle Durchlaufzeit. Diese Kombination aus agilem Development und weitestgehend automatisierten Operationen wird mit den jeweiligen Anfangsbuchstaben zum Kunstbegriff „DevOps“ kombiniert.
Das Ziel ist, schneller bessere Software auszuliefern, was schlussendlich zu zufriedeneren Kunden und zuverlässigen Anwendungen führt.

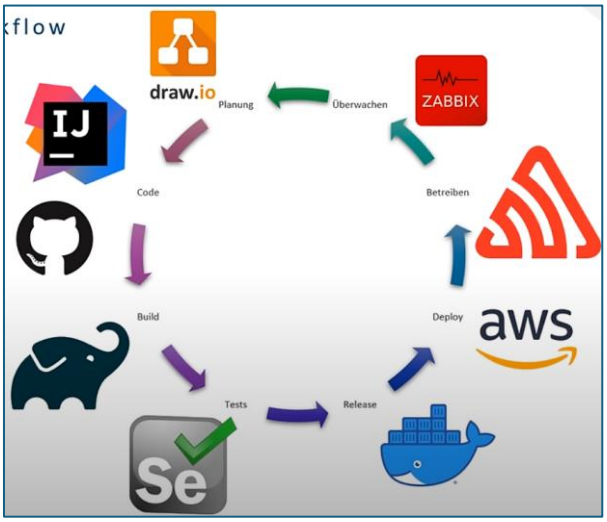
https://www.youtube.com/watch?v=YtQdxrbfYVE&list=PLNmsVeXQZj7pSbRnMtEG5XEdYVwJE5se9&index=1

Vorteile und Nebeneffekte:

- Automatisierung mit Tools und Workflow-Optimierung
- Noch mehr agil (flexibel)

Beispiel: Amazon baut alle 11 Sekunden neue Features in die Plattform ein, das natürlich voll automatisiert. Davor laufen die Tests ebenfalls automatisiert durch.





Tools:

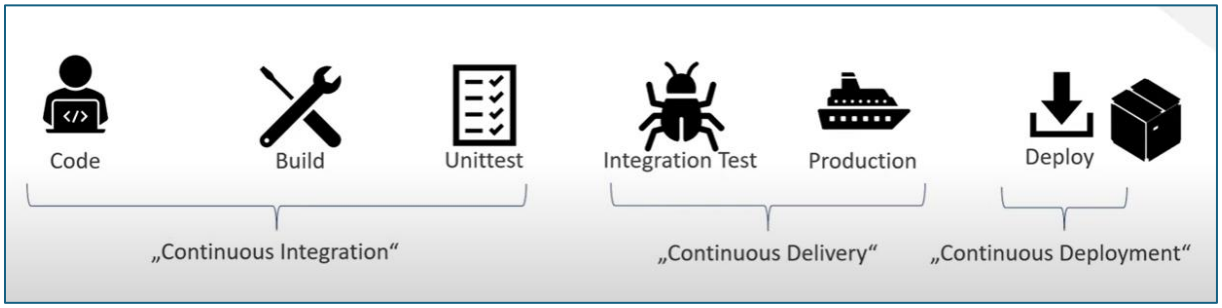
Planung:	draw.io
Code:	VisualStudio-Code, GitHub, GitLab, https://www.jetbrains.com/de-de/idea/
Build	https://maven.apache.org
Testen:	https://www.selenium.dev/
release:	https://www.docker.com/
deploy:	https://aws.amazon.com/de/

Eín Teil von DevOps ist CI/CD:

Continous integration – continous delivery => eine automatisierte Pipeline

Wenn der geschriebene Code automatisiert folgendes durchläuft – mit einem „Knopfdruck“ werden die nachfolgenden Bereiche durchlaufen:

- Build (bei Android z.B. APK-Datei)
- Unittest
- Integration Test
- Production
- Deploy



Das sollte in einem Loop durchlaufen, automatisiert. Dadurch wird die Geschwindigkeit gesteigert.

Testen

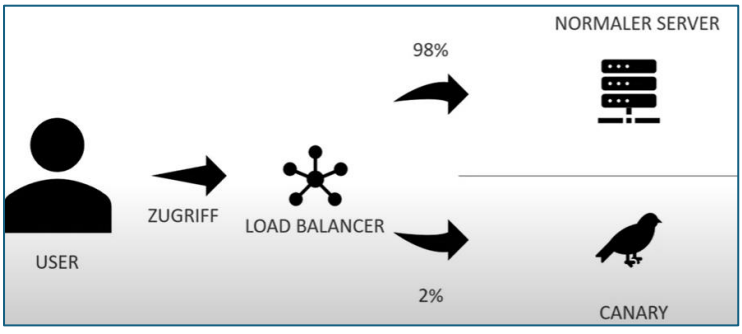
- Unittesting: einzelne Einheiten werden getestet – z.B. auch verbotene Werte testen
- Intergration Testing: das Zusammenspiel funktioniert mit anderen Komponenten
- Performance testing: am besten abends, damit es nicht das Tagesgeschäft blockiert; auch mit „stress testing“ – wie reagiert das System bei vielen Benutzern?
- Security testing: black und white board testing

Deployment

Ausrollen von neuen Features – dem Nutzer / Öffentlichkeit zur Verfügung stellen

z.B. ein Patch, eine neue Version auf der Website, eine neue Version der Android App.

- Canary deployment: (canary = Kanarienvogel, Experimentier-Vögel)
  experimentelle Features ausprobieren – macht Goggle, netflix usw. sehr häufig
  nur ein Teil der User erhält die neuen Features z.B. nur 2% und werden ausgewertet - wie lange bleiben sie auf der Plattform, was klicken sie?



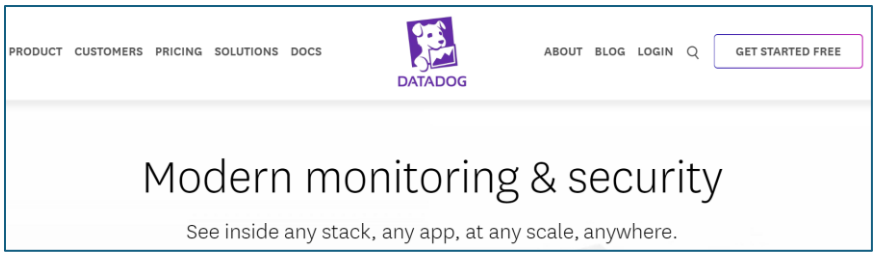
- Immutable deployment = golden image
  Dabei wird das fertige Produkt aus dem „Build-Prozess“ verwendet und direkt gehostet und direkt genutzt.

Monitoring

Hier werden Fragen gestellt und beantwortet wie:

- Welche Performance ist vorhanden?
- Wie funktionieren die einzelnen Komponenten?
- Wie gut läuft das ganze System?
- Wie ist das Nutzerverhalten? Kann man was für den Nutzer oprimieren?
- Security: gibt es Updates?

Logs erstellen und auswerten/analysieren. Die Logs sollten die „5 W“ enthalten, nämlich was, wer, wann, wo und woher. Logs sollten daher nicht zu umfangreich erstellt werden.
Tools:
https://www.datadoghq.com/ - Modern monitoring & security, The integrated platform for monitoring & security



https://www.youtube.com/watch?v=RwtqQWF-tYc&list=PLNmsVeXQZj7pSbRnMtEG5XEdYVwJE5se9&index=11