Hochschule
Albstadt-Sigmaringen

Albstadt-Sigmaringen University

Moderne Softwareentwicklung

Dipl. Ing. Sven Eppler (FH)
sodge IT GmbH

Bringing it all together

Wie wird heute Software Entwickelt?

- **Versionsverwaltung mit git oder vergleichbar**
- **Softwaretests die lokal beim Entwickler ausgeführt werden**
- **Repository-Hosting zum Zusammenführen/Verwalten des Git Repository**
  - z.B. GitHub, GitLab, Azure DevOps
- **Continuous Integration Pipeline**
  - z.B. GitHub Actions, GitLab CI, Jenkins, TravisCI, CircleCI
- **Continuous Deployment Pipeline**
  - z.B. GitHub Actions, GitLab CI, Jenkins, TravisCI, CircleCI

Wie wird heute Software Entwickelt?

- **Qualitätssicherung durch statische Codeanalyse**
  - z.B. ESLint, cpplint
  - Code Reviews
- **Verwendung von Agilen Methoden**
  - z.B. Scrum
- **Verwendung von Feature-Branches** -> z.B. GitFlow
- **Security/Vulnerability Scanning**
  - z.B. Fuzzing
- **Emitteln der Testabdeckung**

Testing, Testing, Testing

Softwaretests

- **Menschen machen Fehler, Menschen schreiben Code, ergo hat Code Fehler**
- **Die einfachste Motivation von Softwaretests ist es, diese Fehler zu finden**
- **Darüber hinaus gibt es Tests die aber auch z.B. Verhalten oder Erwartungen überprüfen**
- **Softwaretests müssen automatisierbar sein.**
  - Denn wenn der Mensch sie ausführen muss, dann vergisst er das zu tun. ;)
- **Wichtigster Baustein für Software-Qualität!**

UnitTesting

- **Testet einzelne Unit in Isolation**
  - Eine Unit: Eine konkrete Klasse mit ihren Methoden
- **Abhängigkeiten werden simuliert (Mock)**
- **Alle Methoden und Features der Unit sollten durch Tests abgedeckt sein**
- **Festzurren des Verhaltens**
  - z.B. "Methode erzeugt eine Exception bei falschen Input"
- **UnitTests sind leichtgewichtig und schnell**

IntegrationTesting

- **Testet eine zusammengehörende Gruppe von Units als eine Einheit**
  - z.B. Teil des Gesamtsystems wie „Buchungslogik“
- **Stellt das funktionale Zusammenspiel dieser Units untereinander sicher**
- **Nicht jeder Edge-Case den jede einzelne Unit abbildet, wird im Integration Test erneut getestet**
- **Abhängigkeiten außerhalb der Gruppe werden simuliert (mock) z.B. Datenbanken, externe APIs, etc.**
- **Festzurren von Schnittstellen**
- **IntegrationTests sind deutlich "fetter" als UnitTests**

SytemTesting

- **Testet das Gesamtsystem mit alle Abhängigkeiten als echte Dienste**
  - z.B. Datenbanken, externe WebAPIs
- **Der SystemTest simuliert i.d.R. die normale Verwendung der Anwendung**
- **SystemTests sind verhältnismäßig schwerfällig und meist auch verhältnismäßig langsam**
  - SystemTest haben in der Realität schnell eine Laufzeit von 30 Minuten oder zum teil Stunden oder sogar Tage
  - Daher werden diese oft auch nur nightly oder weekly ausgeführt

SmokeTesting

- Test der zwischen Build und Push ins Repository lokal ausgeführt wird
- Stellt rudimentär sicher, dass die Anwendung nicht sofort „in Flammen“ aufgeht

AcceptanceTesting

- **Testet User-Zentrierte Anforderungen die als sog. „Akzeptanzkriterien“ formuliert werden**
  - Kriterien wie "Beim klicken auf Speichern, wird das Dokument gespeichert und ein Fortschrittsbalken angezeigt"
- **Akzeptanztests liegen bei Aufwand und Komplexität nahe an SystemTests**

MutationTesting

- MutationsTests sollen Lücken in der Spezifizierung der Unit-/Integrationstests auffinden.
- Damit werden im wesentlichen die Tests getestet
- Dabei wird der Quellcode automatisiert verändert
  - z.B. wird aus einem „==“ Vergleich ein „!=“ Vergleich gemacht
- Danach werden mit dem veränderten Quellcode (dem sog. „Mutant“) die Tests ausgeführt.
- Laufen die Tests jetzt erfolgreich durch, ist eine Lücke in der Spezifikation gefunden

Testing mit Node.js

Mocha Testsuite

- **Mocha ist eine beliebte Testsuite für Node.js**
  - https://mochajs.org
- **Mocha macht:**
  - Entdeckt die Tests im Projekt
  - Führt die Tests aus
  - Sammelt die Testergebnisse zusammen
  - Liefert eine Gesamtergebnis
- **Ausführen von Mocha in einem Node.js Projekt:**
  - npx mocha

Mocha Testsuite

- Standardmäßig nutzt Mocha die sog. „Behavior Driven Development“-Syntax
- Unterstützt aber auch andere Syntax-Arten
- Mocha liefert auch eine „Assert“ Bibliothek zur Überprüfung von Erwartungen
  - Assert.strictEqual(1,2)

Chai Assertion Library

- Chai ist ein Framework das sich auf das überprüfen von Behauptungen spezialisiert hat
  - https://www.chaijs.com
- Ziel dabei ist es, die Testfälle insgesamt möglichst „natürlichsprachlich“ zu definieren

```
let foo = 'Hochschule';
foo.should.be.a('string');
foo.should.equal('Hochschule');
foo.should.have.lengthOf(10);
```

Live-Beispiel: ContactsAPI

- **Zur besseren Demonstration von Tests und späterer Integration mit „Continuous Integration“ befinden sich die Beispiele in einem extra Repository**
  - https://github.com/ghandmann/ContactsAPI

Continuous Integration

Continuous Integration (CI)

- **In einer CI Pipeline sollen immer wieder und automatisiert (kontinuierlich) bestimmte Prozesse ablaufen**
  - Projekt wir automatisch und kontinuierlich gebaut
  - Projekt wir automatisch und kontinuierlich überprüft (statische code analyse, linting)
  - Projekt wir automatisch und kontinuierlich getestet (ausführen der verschiedenen Tests)
- **Dadurch soll der Zustand des Codes immer „im grünen Bereich“ bleiben**

Continuous Deployment

Continuous Deployment (CD)

- **In einer CD Pipeline sollen immer wieder und automatisiert (kontinuierlich) bestimmte Prozesse ablaufen**
  - Erzeugen von Container-Images
  - Erzeugen von Release-Bundles (z.B. Zip-Archive)
  - Erzeugen von ReleaseNotes
- **Sofern sinnvoll (z.B. bei WebAnwendungen) wird automatisch und kontinuierlich auf einer Zielplattform ausgerollt**

DevOps FLow

- Kombination aus „Developer“ und „Operations"
- Software-Entwickler die gleichzeitig die CI/CD Pipeline managen
- Neue Sparte: DevSecOps für DevOps mit security background

CI/CD mit GitHub Actions

- Da sich im Vorlesungs-Repository viele verschiedene Anwendungen befinden, ist eine Integration in GitHub-Actions unnötig kompliziert. Daher finden Sie eine Beispielanwendung im Repository
  - https://github.com/ghandmann/ContactsAPI