

# Continuous Integration

# INTEGRATIONS PROBLEME

# INTEGRATION

## Integration

Zusammenfügen von mehreren Komponenten zu einer Software

- **FALSCH**: dann würde es eher *Continuous Assembly* heißen

# INTEGRATION

## Integration

Zusammenfügen der (lokalen) Entwicklung  
mehrerer Entwicklungszweige

# INTEGRATION

Wenn mehrere Entwickler parallel am gleichen Code arbeiten:

Wie stellen wir sicher, dass die Software, die bisher jeder nur lokal erstellt hat, auch funktioniert wenn alle Änderungen zusammenfließen?

# PROBLEM BEREICHE

## Merge-Konflikte

Entwickler haben **gleichzeitig** die gleiche Datei bearbeitet

## Kompilier-Konflikte

keine Merge-Konflikte, aber die **merged** Codebasis kompiliert nicht

## Test-Konflikte

keine Merge-Konflikte, keine Kompilier-Konflikte, aber die **Tests** laufen nicht mehr erfolgreich

# INTEGRATION

## Integration

Code-Basis zweier Entwickler ineinander integrieren um alle Arten von Konflikten zu identifizieren.

# AUTOMATISIERUNGS PROBLEME



# PUR

## HelloWorldApp.java

```
class HelloWorldApp {  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
    }  
}
```

## Kompilieren

```
javac HelloWorldApp.java
```

## Ausführen

```
java -cp . HelloWorldApp  
java -classpath . HelloWorldApp
```

# PUR + BIBLIOTHEK

## HelloWorldApp.java

```
import org.apache.commons.lang3.StringUtils;  
class HelloWorldApp {  
    public static void main(String[] args) {  
        String msg = "Hello World!";  
        msg = StringUtils.substring(msg, 6)  
        System.out.println(msg);  
    }  
}
```

## Kompilieren

```
$ javac -cp lib/commons-lang.jar HelloWorldApp.java
```

## Ausführen

```
$ java -cp lib/commons-lang.jar;. HelloWorldApp
```

# KOMPLIZIERTER

- Woher kommt `commons-lang.jar`?
- Welche Version von `commons-lang.jar` wird verwendet?
- Wie viele Bibliotheken verwendet `dhbw-paingroup` insgesamt?
- Antwort: [112](#)

# AUTOMATISIERUNG

# KOMPILIEREN

Kommandozeile:

```
javac src/main/java/net/kleinschmager/dhbw/tfe15/paingroup/Paingroun  
src/main/java/net/kleinschmager/dhbw/tfe15/paingroup/ui/MainUI.java  
src/main/java/net/kleinschmager/dhbw/tfe15/paingroup/ui/views/Member  
src/main/java/net/kleinschmager/dhbw/tfe15/paingroup/persistence/mod  
src/main/java/net/kleinschmager/dhbw/tfe15/paingroup/persistence/rep
```

IDE: Menü *Projekt* > *Bereinigen*

# VERPACKEN

## Kommandozeile

```
jar cvmf painground.jar src/main/java/net/kleinschmager/dhbw/tfe15/pa  
src/main/java/net/kleinschmager/dhbw/tfe15/painground/ui/MainUI.java  
src/main/java/net/kleinschmager/dhbw/tfe15/painground/ui/views/Member  
src/main/java/net/kleinschmager/dhbw/tfe15/painground/persistence/mod  
src/main/java/net/kleinschmager/dhbw/tfe15/painground/persistence/rep
```

## IDE

- siehe nächste Folie



# NACHTEILE

## KOMMANDOZEILE

- nicht übersichtlich
- unkomfortabel
- Abhängig von Umgebung
  - javac version
  - Bibliotheken
- *Works on my machine*

## IDE

- Abhängig von Umgebung
  - javac version
  - Bibliotheken
  - IDE Konfiguration
- *Works on my machine*



# WEITERE AUFGABEN

- Testen
- Dokumentation erzeugen
  - Word zu PDF?
  - *xyz* zu HTML?
- Upload zum Kunden
- Bereitstellen DEMO System

# LÖSUNG: AUTOMATISIERUNG

- Build-Tools
  - Ant | Maven | Gradle | CMake
- Continuous Integration
  - Mindset
- Continuous Integration Tools
  - Jenkins
  - Travis-CI
  - Team Foundation Server

# CONTINUOUS INTEGRATION

# MOTIVATION

*In software, when something is painful,  
the way to reduce the pain is to do it  
more frequently, not less.*

— David Farley

# ELEMENTE

1. Code (und Konfiguration) stehen unter Versionsverwaltung
2. Build-Prozess ist automatisiert
3. Regelmäßiges einchecken|commit
  - mind. täglich

# ELEMENTE

4. **Tests** werden gleichzeitig entwickelt (als Code)
  - stehen ebenfalls unter Versionsverwaltung
  - am besten im gleichen Repository wie der Code selbst
5. Wichtige Tests sollten **bei jedem commit** ausgeführt werden
  - andere wenigstens regelmäßig, z.B. nächtlich
6. eine **produktionsnahe** Testumgebung steht immer bereit
7. Einfacher Zugriff auf Ergebnisse auch für **Nicht-Entwickler**



# VORTEILE

*Continuous Integration doesn't get rid of bugs, but it does make them dramatically easier to find and remove.*

— Martin Fowler



# VORTEILE

## Continuous Integration

regelmäßiges Kompilieren, Verpacken, Testen,  
Bereitstellen einer Software

# VORTEILE

- Fehler früher finden (Konflikte vermeiden)
- Feedback für das Entwickler-Team
- Feedback für das Qualitäts-Management
- Feedback für die Tester

# CONTINUOUS DELIVERY

*Continuous Delivery is a software development discipline where you build software in such a way that the software can be released to production at any time.*

— Martin Fowler

# CONTINUOUS DELIVERY

*There should be two tasks for a human being to perform to deploy software into a development, test, or production environment: to pick the version and environment and to press the “deploy” button.*

— David Farley