

# Technik und Werkzeuge

Software Engineering & Projektmanagement VO  
(188.410)

Richard Mordinyi  
richard.mordinyi@tuwien.ac.at

# Agenda

- Source Code Management
- Build Management
- Komponentenorientierte Software-Entwicklung
- Kommunikation im Team

# Sourcecode-Management (SCM) Systeme

- Motivation
  - (verteilte) Teamarbeit → Parallele Projektentwicklung
  - Verwaltung von Artefakten
    - Source Code, Dokumentation, Diagramme...
  - Parallele Entwicklungslinien

# Sourcecode-Management (SCM) Systeme

- Motivation
  - (verteilte) Teamarbeit → Parallele Projektentwicklung
  - Verwaltung von Artefakten
    - Source Code, Dokumentation, Diagramme...
  - Parallele Entwicklungslinien
- Änderungen transparent machen
- Entwicklungsgeschichte nachvollziehen
- Zugriff auf ältere Versionen ermöglichen
- Gleichzeitige Entwicklung mehrerer Zweige

# Versionierungssysteme

- Entwicklungsgeschichte
  - Revision-Control-Systeme (RCS)
    - Auf Basis einzelner Dateien
  - Concurrent-Version System (CVS)
    - Zentrales, Server-basiertes Repository
  - Verteiltes SCM
    - jeder Entwickler hat sein eigenes Repository
    - mit jedem beliebigen anderen Repository abgleichbar

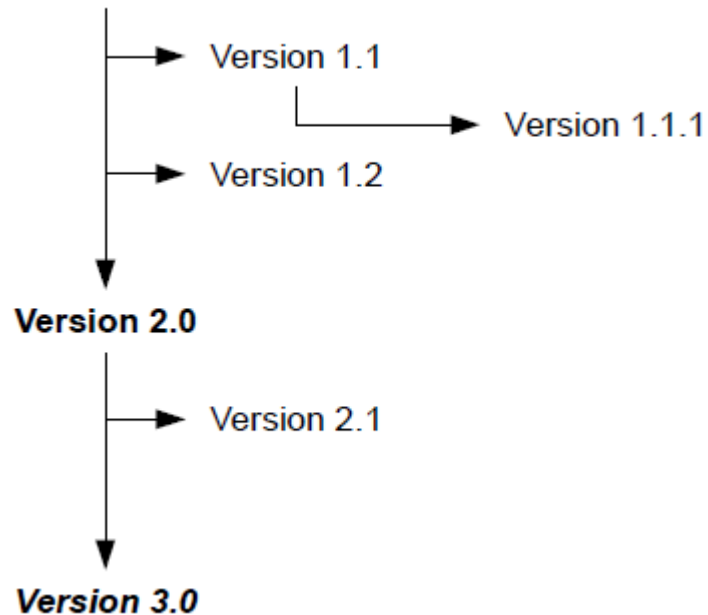
Zentrale Systeme	CVS, Subversion (SVN)
Verteilte Systeme	Git, Mercurial, BitKeeper

# Versionierungssysteme

- Terminologie
  - Check-out
    - Lokale Kopie einer Version anlegen
  - Check-in (Commit)
    - Änderungen seit dem letzten Update (changeset) in das Repository übernehmen
    - Commit-Message
  - Tagging
    - Markieren eines bestimmten Zustands des Systems
  - Merging
    - Änderungen aus einem Branch in einen anderen übernehmen

# Parallele Entwicklungspfade

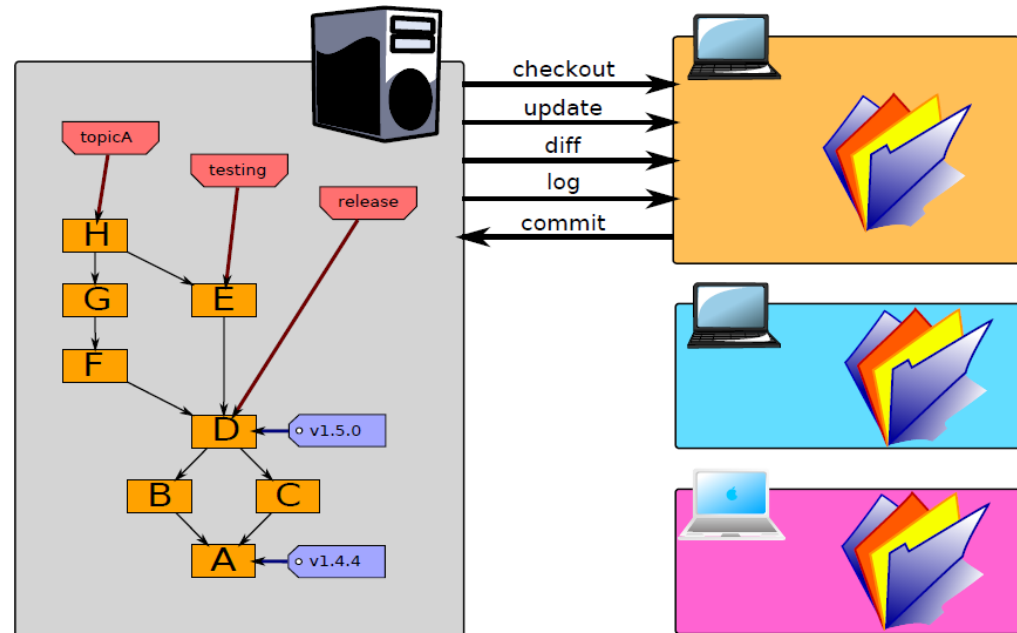
**Version 1.0**



- Branching
  - Abspaltung einer Version
  - Entwickeln in parallelen Strängen
- Wartungsaufgaben
- Experimentelle Features

# Zentralisierte Systeme

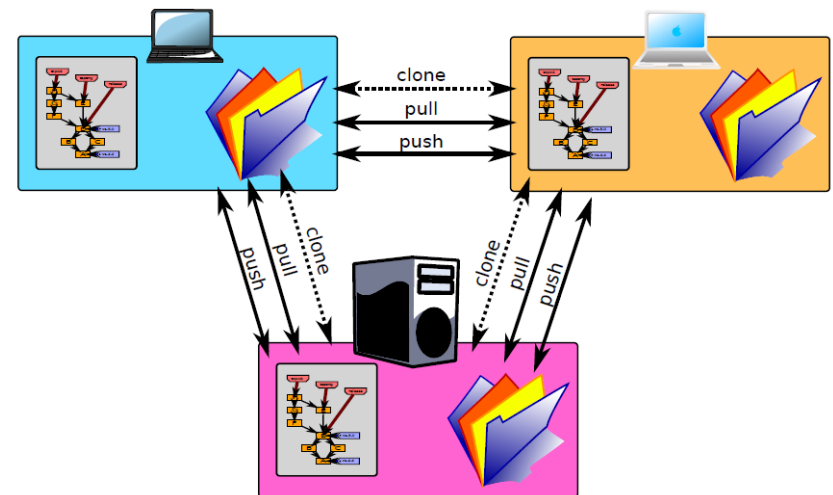
- Server speichert Versionen
  - Single-Point of Failure
  - Flaschenhals
- Operationen erfordern Netzwerkverkehr





# Verteilte Systeme

- Repository wird geklont
  - Große Datenmengen
  - Viele kleine Dateien
- Projektmanager mit „Main-Repository“
  - Pull-requests
- Jede Komponente in einer eigenen Repository
  - Minimierung der zu kopierenden Datenmenge
- Lokale Operationen
  - Lokale Versionierung



# Zentral vs. Verteilte Systeme

- Empfehlungen
  - Code sauber und konsistent halten
    - Kompilierbar
    - Alle Tests positiv
    - Regelmäßig commiten
  - Nichts einchecken, das nicht erzeugt werden kann
- Performance
- Verwaltungsaufwand
- Scope
- Branching
- Offline Modus

# Umgang mit Konflikten

- Teamarbeit
  - Mehrere Personen arbeiten an derselben Datei
  - Konfliktpotential
- Strategien
  - Locking
  - Merging
- Arbeitsablauf - SVN
  - Update
  - Lokale Änderungen durchführen
  - Update
    - Änderungen remote und lokal
    - Merge
  - Commit
- Arbeitsablauf – Git
  - Kopie des Haupt-Repository
  - Lokale Änderungen durchführen
  - Lokale Commits
  - Merge
    - Push
    - Pull Request

# Arbeitsabläufe im traditionellen Software Engineering

- Vielzahl an sich wiederholenden Aufgaben
- Manuelle Ausführung
  - Lästig
  - Fehleranfällig
  - Langsam
  - Unmöglich
- Software-Entwicklung sollte sein
  - Deterministisch
  - Fehlerfrei
  - Effizient

# Was sollte automatisiert werden?

- Code Quality Checks
  - Validierung von Source Code
  - Code-Generierung
  - Kompilierung
  - Testausführung
  - Informationssammlung für Reporting/Dokumentation
  - Packaging
  - Deployment
  - Dependency Management
- 
- Wie oft?

***"The most fundamental part of the daily build is the 'daily' part. . . . Treat the daily build as the heartbeat of the project. If there's no heartbeat, the project is dead."***

**James McCarthy**

# Maven - Build Management System

- Build Tool
  - Kompilierte Applikation automatisch erzeugen
- Vereinfacht Management von Java Projekten
- Dependency Management
- Deployment Tool
- Dokumentation aus javadoc
  
- maven
  - Best-Practices
  - Convention over Configuration
    - Archetypes
  - Project Object Model (POM)
  
- Ant
  - Make
  - Build Xml
  - Definition von Tasks

# Convention over Configuration



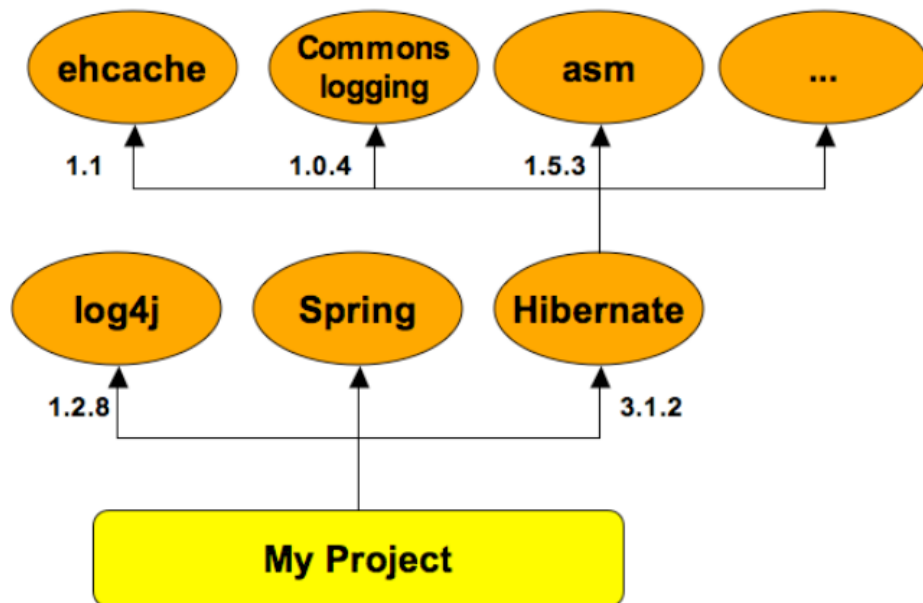
```

<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.mycompany</groupId>
  <artifactId>my-app</artifactId>
  <packaging>jar</packaging>
  <name>Sample application</name>

  <dependencies>
    ...
  </dependencies>
</project>

```

# Dependency Management



- Entwicklung im Team
  - Konsistenter Zustand
  - Entwicklung, Tests, Auslieferung
- Aktualisierung
  - Inkompatibilität
- transitive Abhängigkeiten
  - Versions-Konflikte



# Reporting und Dokumentation

- Generelle Projektinformationen
  - Version, Name, Beschreibung
  - Beitragende
  - Lizenz
  - Entwicklungswerkzeuge
- Source Code (javadoc)
- Code Quality Reports
  - Test Reports
  - Checkstyle
  - Component Dependency Reports
- Dokumente
  - Architekturbeschreibung
  - Anwenderhandbuch
  - Website

# Maven Build-Lifecycle

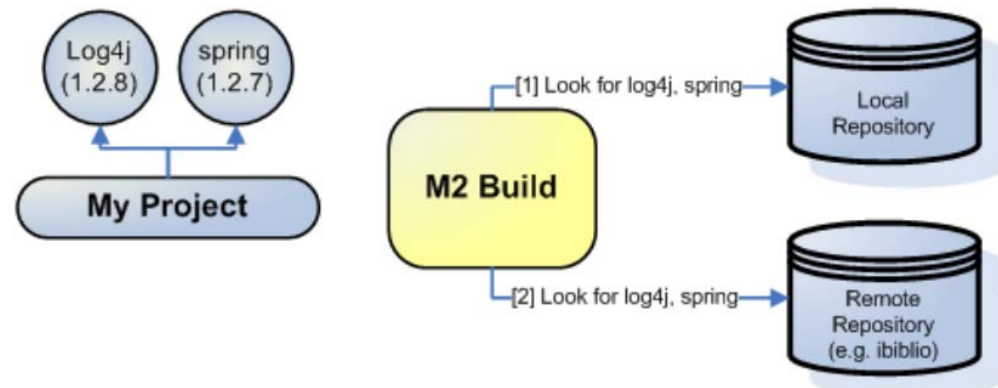
<b>validate</b>	validate the project is correct and all necessary information is available.
<b>generate-sources</b>	generate any source code for inclusion in compilation.
<b>process-sources</b>	process the source code, for example to filter any values.
<b>generate-resources</b>	generate resources for inclusion in the package.
<b>process-resources</b>	copy and process the resources into the destination directory, ready for packaging.
<b>compile</b>	
<b>process-classes</b>	post-process the generated files from compilation, for example to do bytecode enhancement on Java classes.
<b>generate-test-sources</b>	generate any test source code for inclusion in compilation.
<b>process-test-sources</b>	process the test source code, for example to filter any values.
<b>generate-test-resources</b>	create resources for testing.
<b>process-test-resources</b>	copy and process the resources into the test destination directory.

# Maven Build-Lifecycle

<b>test-compile</b>	compile the test source code into the test destination directory
<b>test</b>	run tests using a suitable unit testing framework. These tests should not require the code be packaged or deployed.
<b>package</b>	take the compiled code and package it in its distributable format, such as a JAR.
<b>pre-integration-test</b>	perform actions required before integration tests are executed. This may involve things such as setting up the required environment.
<b>integration-test</b>	process and deploy the package if necessary into an environment where integration tests can be run.
<b>post-integration-test</b>	perform actions required after integration tests have been executed. This may include cleaning up the environment.
<b>verify</b>	run any checks to verify the package is valid and meets quality criteria.
<b>install</b>	install the package into the local repository, for use as a dependency in other projects locally.
<b>deploy</b>	done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects.

# Dependency Management in Maven

- Abhängigkeiten in pom.xml deklarieren
  - Name, Version
- Transitive Abhängigkeiten müssen nicht deklariert werden
- pom.xml-Datei im Sourcecode-Management-System ablegen
  - Änderungen der Abhängigkeiten nachvollziehbar
- Maven legt Bibliotheken aller Projekte in einem zentralen Verzeichnis ab
- Deklarierte Abhängigkeiten prüfen und ggf. herunterladen
- Transitive Abhängigkeiten prüfen und ggf. herunterladen
- Klassenpfad für Compile- und Test-Lauf erstellen



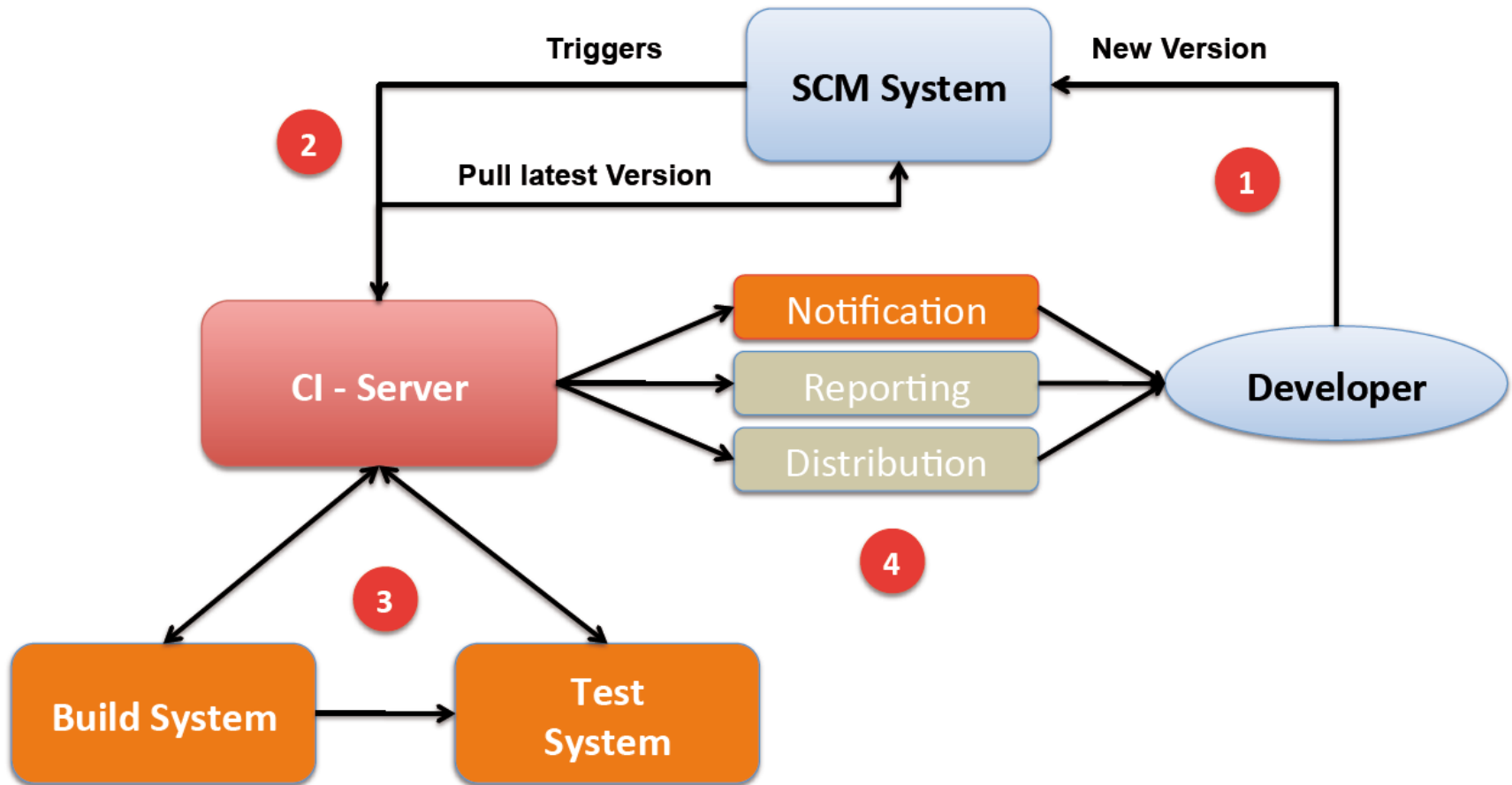
# Continuous Integration

- Vorteile einer Build-Automatisierung
  - Projekt schnell mit Archetypen aufsetzen
  - Projekt ist portabel
  - Abhängigkeiten leichter darstellbar und auflösbar
  - Verbesserung der Projekt-Qualität durch Integration automatisierter Tests
  - Automatisiertes Reporting
- Einsatz im Continuous Integration Kontext
- Best-practices Aspekte
  - ein Sourcecode-Repository
  - Automatisierter Build
  - Automatisierte Testabläufe
  - Tägliche Commits der Entwickler
  - Schneller Build
  - Build auf einem neutralen Rechner
    - daily builds, snapshots...
  - Automatisiertes Deployment
  - Transparenz des Entwicklungsprozesses

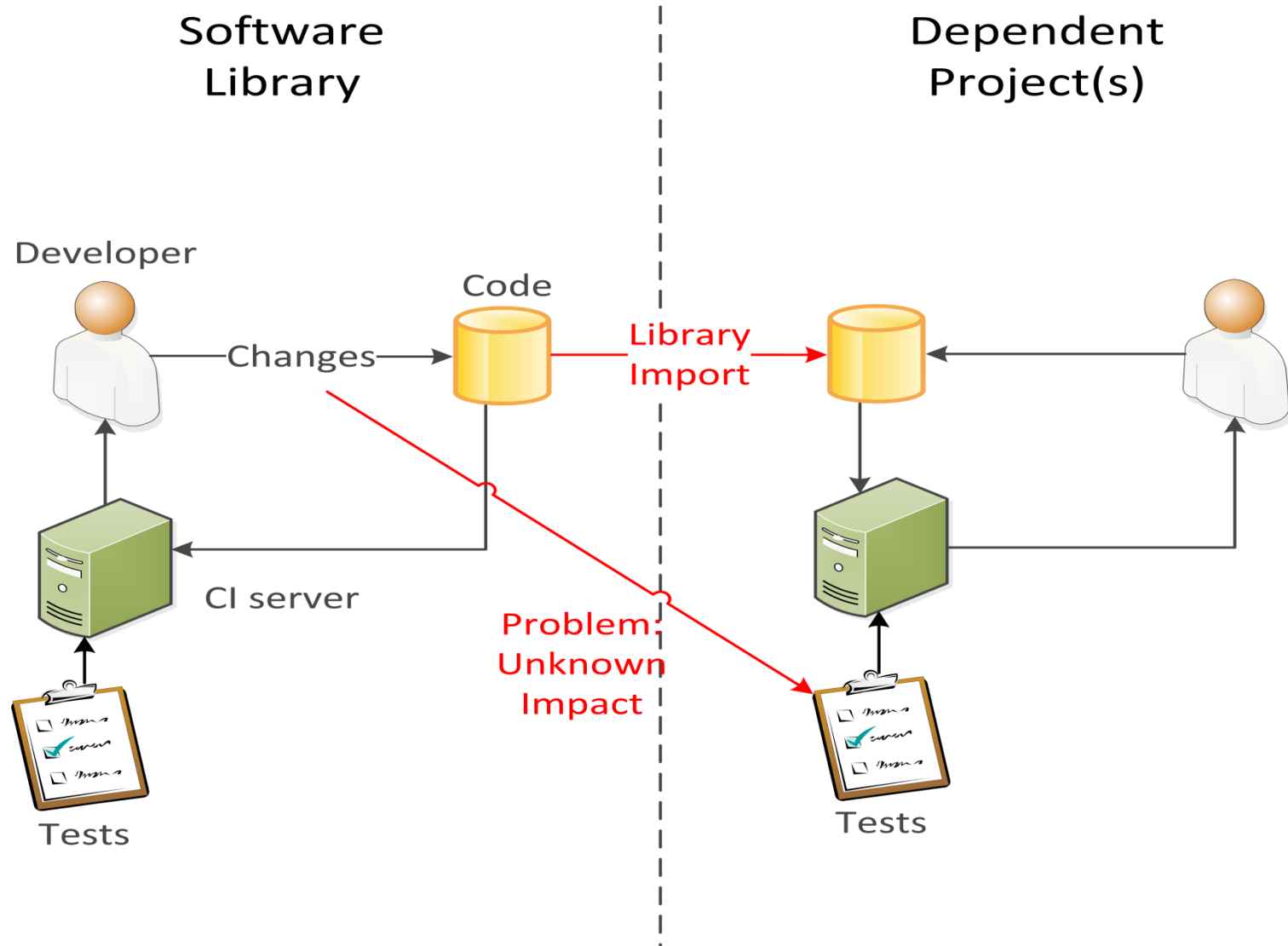
# Continuous Integration Werkzeuge

- Build-Automatisierungswerkzeuge wie Maven sind Entwicklerorientiert
- Continuous Integration Werkzeuge (Apache Continuum, Hudson, OpenCIT) unterstützen Server-basierte Integration und Ausführung von Tests
- Automatisierung beinhaltet
  - Ereignis oder Zeitgesteuerter Abruf
  - Verwendung von Build Werkzeugen
  - Ausführung von Tests
  - Erstellen von Berichten
  - Verschicken von Mitteilungen

# Continuous Integration Arbeitsablauf

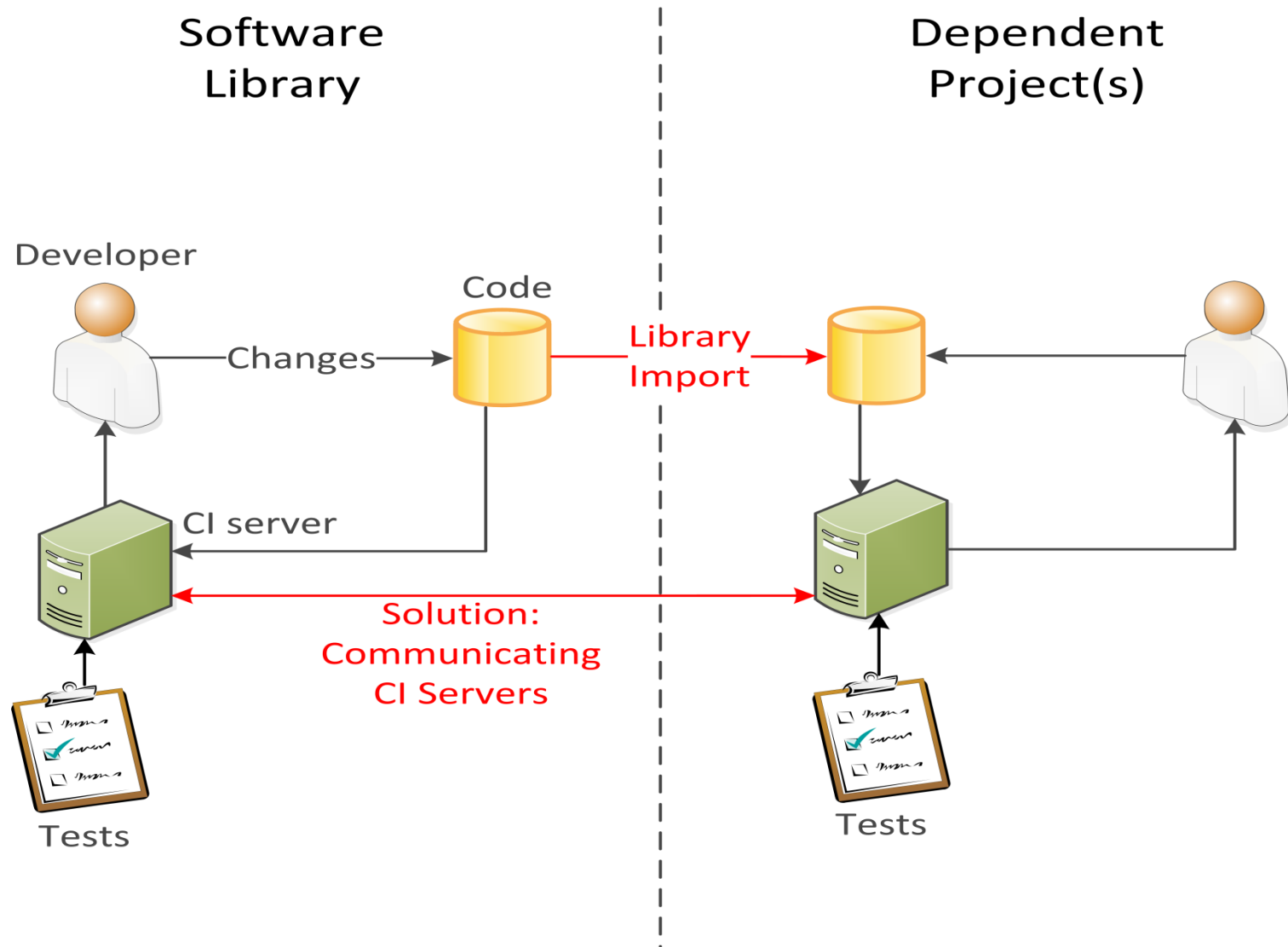


# Continuous Integration - Limitierungen





# Continuous Integration mit OpenCIT



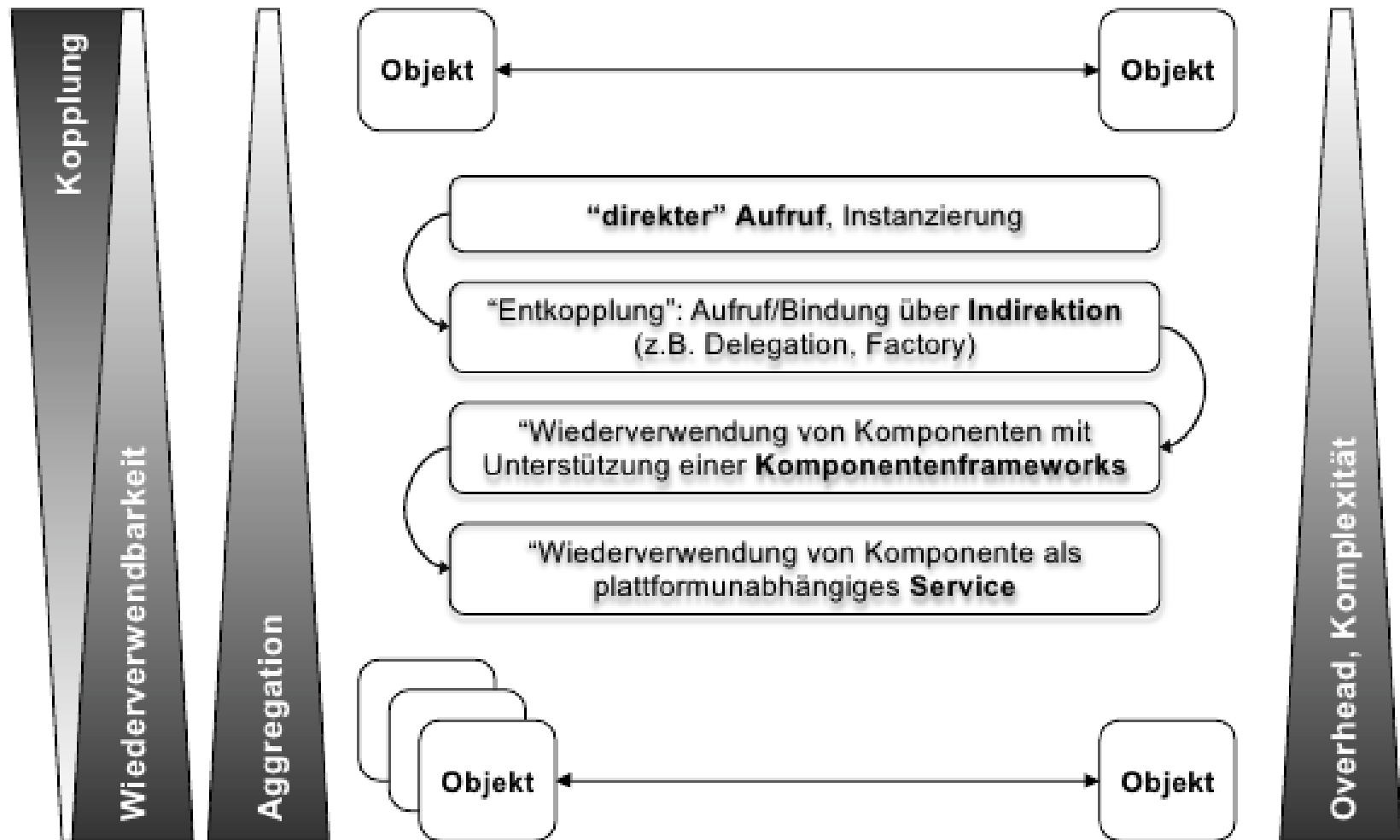
# Zusammenfassung

- Sourcecode Management Systeme
  - Effiziente Zusammenarbeit in verteilter Software Entwicklung
  
- Build Management Systeme
  - Minimierung von Aufwänden durch hohen Grad an Automatisierung

# References

- Vortrag Linus Torvald über Git
  - <http://www.youtube.com/watch?v=4XpnKHJAok8>
- <http://git-scm.com/>
- <http://excess.org/article/2008/07/ogre-git-tutorial/>
- <http://subversion.tigris.org/>
- <http://maven.apache.org/>
- <http://opencit.openengsb.org/>

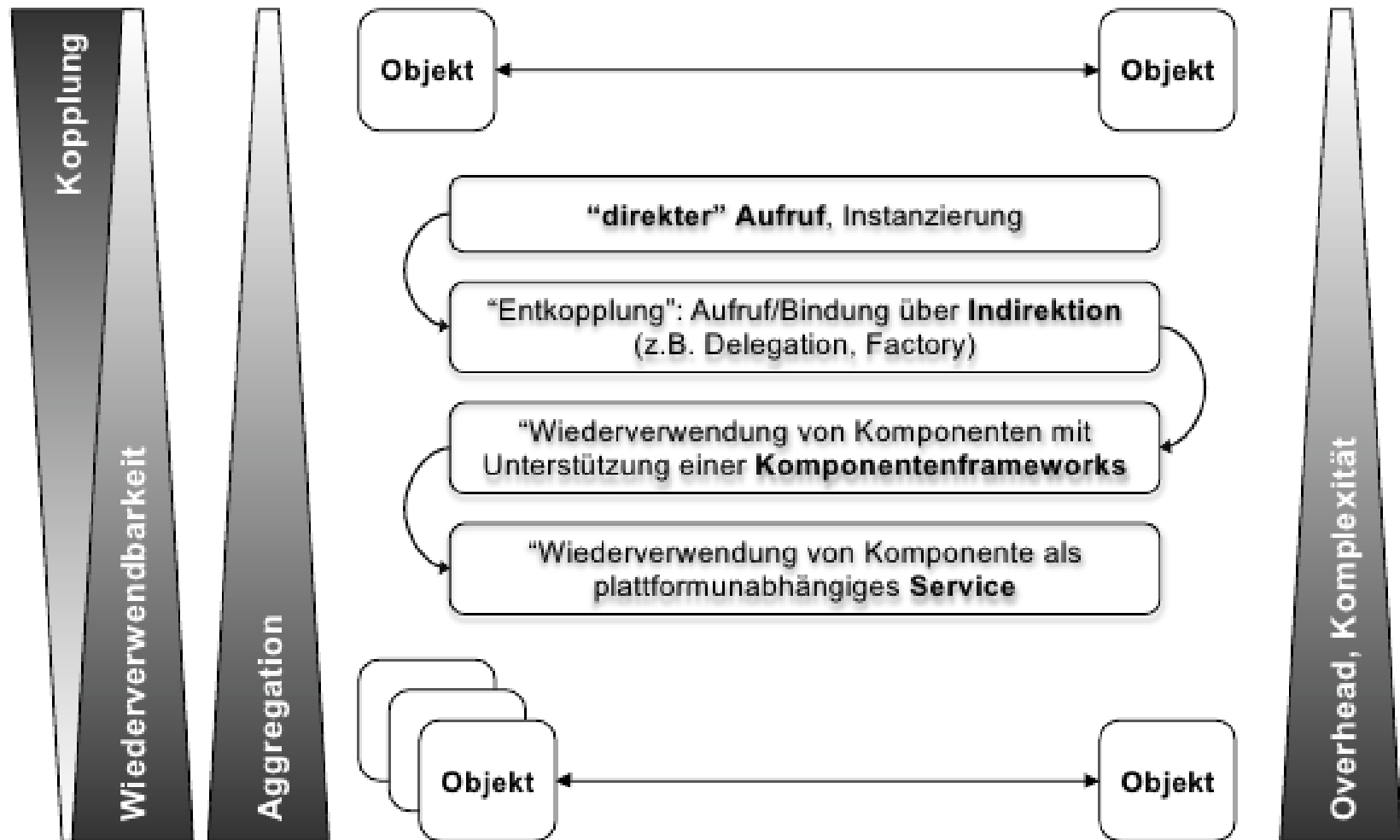
# Komponentenorientierte Software-Entwicklung



# Begriffe

- Komponente
  - klare, stabile Schnittstelle
  - höhere Granularität als eine Klasse
  - Wiederverwendbarkeit
  - locker verbunden
- Service
  - klar definierte Schnittstelle
  - plattformunabhängig
  - über ein Netzwerk angeboten
- Framework
  - Wiederverwendbarkeit
  - Rahmenbedingungen für Komponenten (z.B. Persistenz-Framework)

# Komponentenorientierte Software-Entwicklung

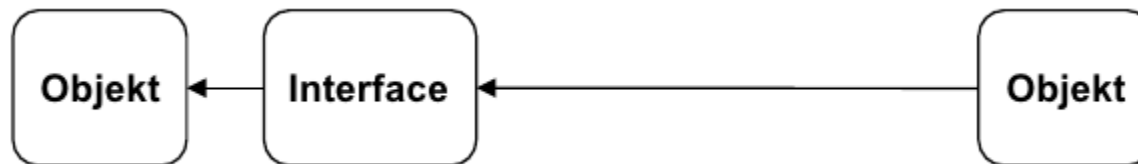


# Direkter Aufruf

```
public class Main {  
    public static void main(String[] args) {  
        Handler handler = new Handler();  
        handler.notify();  
    }  
}
```

- leicht zu implementieren
- Binding beim Kompilieren
- Starke Kopplung

# Interface





# Interface

```

1 | public interface TextImport {
2 |     public Document read (String filename);
3 | }

```

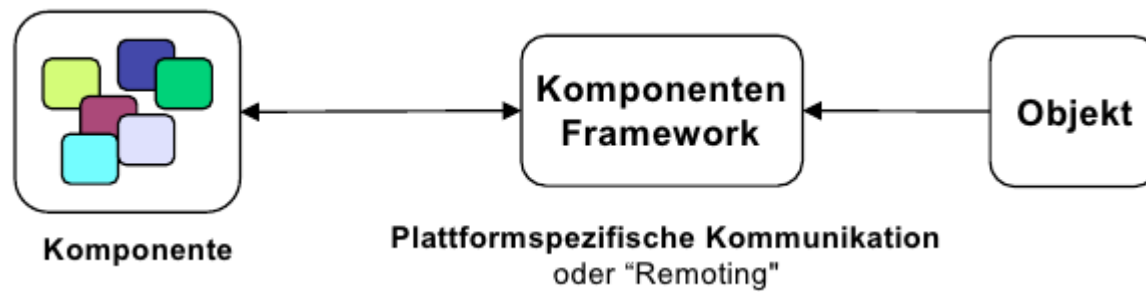
```

1 | class A {
2 |     TextImport ti = new HtmlImport();
3 |     public processDocument (String filename) {
4 |         Document doc = ti.read(filename);
5 |         ...
6 |     }
7 | }

```

- Contract
- Einfacheres Austauschen der Implementierung
- Binding beim Kompilieren

# Komponenten



# Komponenten

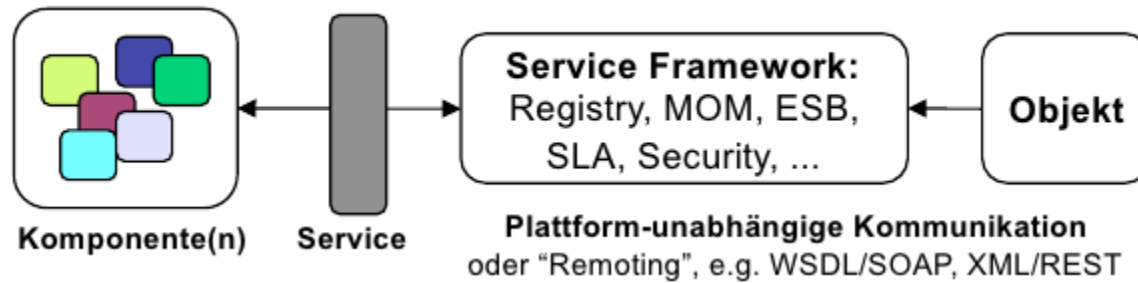
```

1  class KlasseA {
2      TextImport ti;
3      public KlasseA() {
4          // pseudocode:
5          ti = componentFramework.
6              getComponent ("htmlimport");
7      }
8      public processDocument (String filename) {
9          Document doc = ti.read(filename);
10         ...
11     }
12 }

```

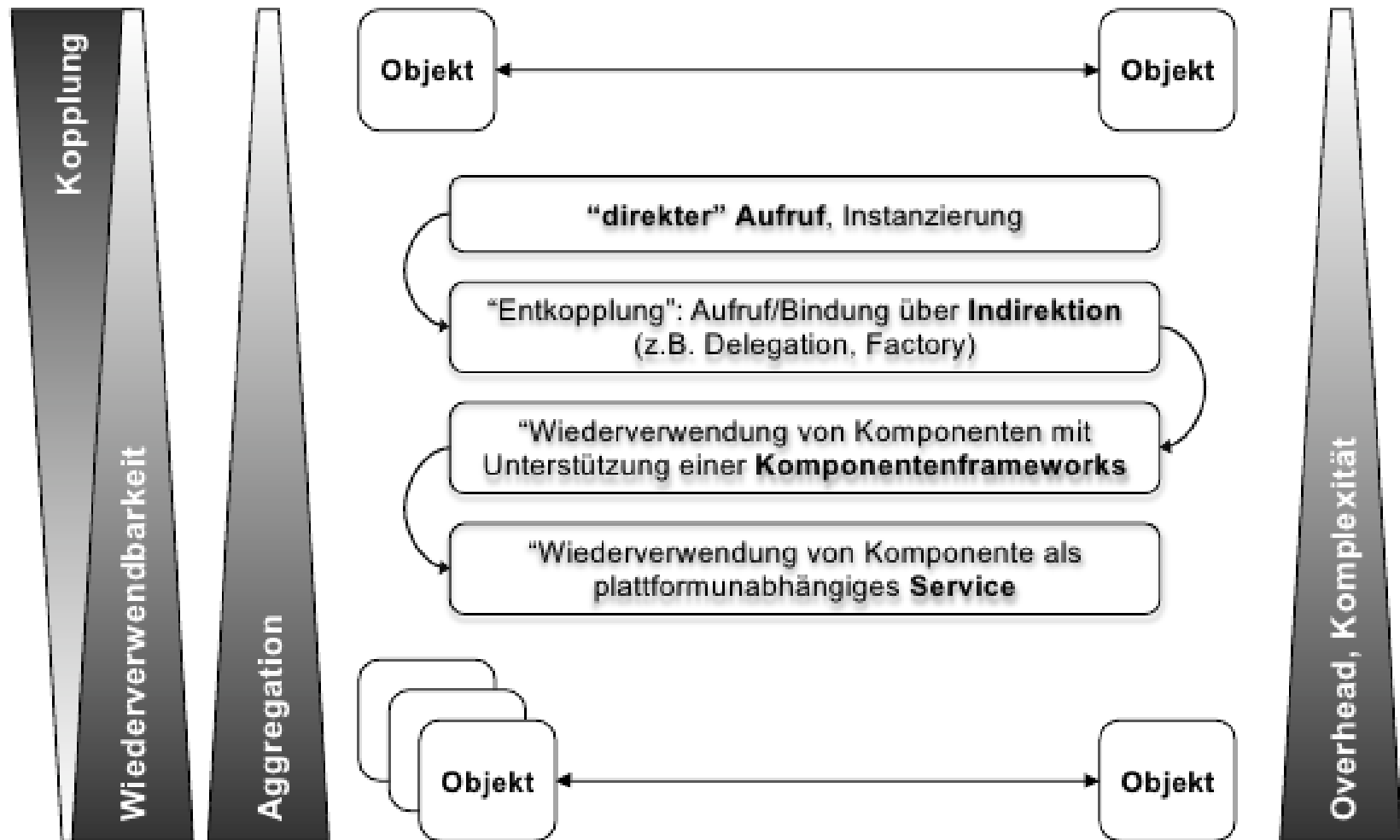
- Austauschen der Implementierung ohne Änderung im Sourcecode
- Verantwortlichkeit bei der Klasse selbst

# Services



- Kommunikation über Systemgrenzen
- Webservices (WSDL und SOAP)

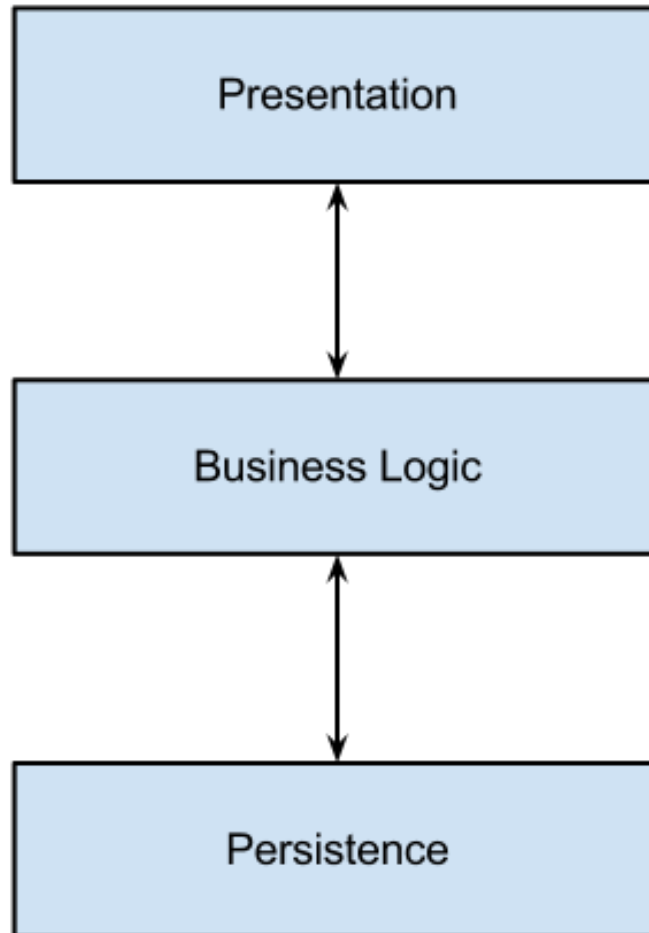
# Komponentenorientierte Software-Entwicklung



# Verbinden von Komponenten

- Software besteht aus mehreren Komponenten
- Architektur bestimmt Verbindungen und Zusammenspiel zwischen Komponenten
- Grad der Entkopplung
- Komplexität

# Beispiel Anwendung



# Beispiel Anwendung – Variante 1

```
public class MyPersistence {
    | private MyBusinessLogic businessLogic = new MyBusinessLogic();
    | // ...
}
public class MyPresentation {
    | private MyBusinessLogic businessLogic = new MyBusinessLogic();
    | // ...
}
public class MyBusinessLogic {
    | private MyPersistence persistence = new MyPersistence("path/to/db");
    | private MyPresentation presentation = new MyPresentation(300, 200);
    | // ...
}
```



# Beispiel Anwendung – Variante 2

```
public static void main(String[] args) {
    | IPersistence persistence = new MyPersistence("path/to/db");
    | IPresentation presentation = new MyPresentation(300, 200);
    |
    | IBusinessLogic businessLogic =
    | | | | new MyBusinessLogic(persistence, presentation);
    |
    | persistence.setBusinessLogic(businessLogic);
    | presentation.setBusinessLogic(businessLogic);
    |
}
```

# Inversion of Control (IoC)

- Abhängigkeiten wird von einem Container verwaltet
- Komponenten wissen nichts darüber
- Abhängigkeiten werden in Komponenten injiziert  
→ „Dependency Injection“ (DI)
- Frameworks mit Hollywood Prinzip:
  - *"Don't call us, we'll call you"*

```
public static void main(String[] args) {  
  
    IPersistence persistence = new MyPersistence("path/to/db");  
  
    IPresentation presentation = new MyPresentation(300, 200);  
  
    IBusinessLogic businessLogic =  
        new MyBusinessLogic(persistence, presentation);  
  
    persistence.setBusinessLogic(businessLogic);  
    presentation.setBusinessLogic(businessLogic);  
  
}
```

# Beispiel Anwendung – Variante 3

```

public static void main(String[] args) {
    Factory factory = Factory.create();

    IPersistence myPersistence =
        (IPersistence) factory.getComp("persistence");

    IPresentation myPresentation =
        (IPresentation) factory.getComp("presentation");

    MyBusinessLogic myBusinessLogic =
        new MyBusinessLogic(myPersistence, myPresentation);

    myPersistence.setBusinessLogic(myBusinessLogic);
    myPresentation.setBusinessLogic(myBusinessLogic);
}

```

# Beispiel Anwendung – Variante 4 (1/2)

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<beans>
```

```
  <bean class="MyPersistence" id="persistence">
    <property name="path">path/to/db</property>
    <property name="businessLogic" ref="businessLogic" />
  </bean>
```

```
  <bean class="MyPresentation" id="presentation">
    <property name="width">300</property>
    <property name="height">200</property>
    <property name="businessLogic" ref="businessLogic" />
  </bean>
```

```
  <bean class="MyBusinessLogic" id="businessLogic">
    <property name="persistence" ref="persistence" />
    <property name="presentation" ref="presentation" />
  </bean>
```

```
</beans>
```

# Beispiel Anwendung – Variante 4 (2/2)

```
public static void main(String[] args) {  
    Resource res = new FileSystemResource("beans.xml");  
    XmlBeanFactory factory = new XmlBeanFactory(res);  
  
    IPresentation ui =  
        factory.getBean("presentation", IPresentation.class);  
    ui.show();  
}
```

# Vorteile von IoC

- Hohe Wiederverwendbarkeit durch zentrale Verwaltung
- Einfaches Austauschen einer Implementierung
- Verwalten von verschiedenen Konfigurationen
  - dev vs. deploy
- Automatisiertes Verdrahten
- Verteilung von Aufgaben

# IoC Container Implementierungen

## Java

- Spring-Framework
- Google Guice
- Pico-Container
- Apache Aries Blueprint

## .NET

- Unity Framework
- Spring.NET



# Constructor Injection

- Abhängigkeiten durch entsprechenden Konstruktor
- Beispiele: Pico, Guice, (Spring, Aries)
- Definition vollständiger Konfigurationen
- Komplexe Objekte können zu langen Konstruktoren führen
- Benennung der Argumente
- Problem mit einfachen Typen

# Constructor Injection - Limitierungen

```
public class TestBean {

    String id;
    String internalLocation;
    String externalLocation;

    public TestBean(String id, String internalLocation) {
        this.id = id;
        this.internalLocation = internalLocation;
    }

    public TestBean(String id, String externalLocation) {
        this.id = id;
        this.externalLocation = externalLocation;
    }

}
```

# Setter Injection

- Abhängigkeiten durch "setter"-Methode
- Beispiele: Spring, Aries, (Pico)
  
- Oft bevorzugte Variante
- Identifikation von Attributen mittels
  - Namenskonvention (z.B. `setComponent()`)
  - Annotationen

# Konfigurationsdatei

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<beans>
```

```
  <bean class="MyPersistence" id="persistence">
    <property name="path">path/to/db</property>
    <property name="businessLogic" ref="businessLogic" />
  </bean>
```

```
  <bean class="MyPresentation" id="presentation">
    <property name="width">300</property>
    <property name="height">200</property>
    <property name="businessLogic" ref="businessLogic" />
  </bean>
```

```
  <bean class="MyBusinessLogic" id="businessLogic">
    <property name="persistence" ref="persistence" />
    <property name="presentation" ref="presentation" />
  </bean>
```

```
</beans>
```

# Configuration in Code

```
@Configuration
public class AppConfig {

    @Bean
    public IPersistence persistence() {
        MyPersistence myPersistence = new MyPersistence("path/to/db");
        myPersistence.setBusinessLogic(businessLogic());
        return myPersistence;
    }

    @Bean
    public IPresentation presentation() {
        MyPresentation myPresentation = new MyPresentation(300, 200);
        myPresentation.setBusinessLogic(businessLogic());
        return myPresentation;
    }

    @Bean
    public IBusinessLogic businessLogic() {
        return new MyBusinessLogic(persistence(), presentation());
    }
}
```

# Zusammenfassung

- Begriffe: Komponente und Service
- Verwalten von Komponenten
  - Direkt
  - Factory
  - Komponentenframework
- Dependency Injection/Inversion of Control
  - Constructor vs Setter Injection
  - Configuration File vs Code



# Kommunikation im Team

- Synchron
  - Face-to-Face
  - Telefon
  - Instant Messaging, VoIP (XMPP, Skype)
- Asynchron
  - Email
  - Wiki, CMS
  - Mailingliste/Forum
  - Issue-Tracker



# Mailingliste/Forum

- Zentrale Verwaltung von Interessenten
- Diskussionen besser nachvollziehbar
- Mailinglisten für verschiedene Zwecke
  - developer
  - user
  - announce

# Mailingliste Beispiel

Groups
NEW TOPIC

OpenEngSB developer discussion

+1
0

Homepage redesign

By Christoph Gritschenberger - 8 posts - 6 views - updated 9:23 AM (3 hours ago)

Multiple EDB's

By Felix Mayerhuber - 5 posts - 0 views - updated Apr 13 (3 days ago)

[build] CIT workflow report - FAILURE

By openengsb admin - 2 posts - 0 views - updated Apr 12 (3 days ago)

Concepts domains and connectors

By Kristof - 7 posts - 0 views - updated Apr 12 (3 days ago)

OpenEngSB Distribution Release Policy

By Andreas Pieber - 7 posts - 4 views - updated Apr 12 (3 days ago)

[Proposal] Move Issue-branches to own forks

By Christoph Gritschenberger - 3 posts - 4 views - updated Apr 12 (4 days ago)

[build] CIT workflow report - FAILURE

By openengsb admin - 4 posts - 0 views - updated Apr 11 (4 days ago)

[build] CIT workflow report - SUCCESS

By openengsb admin - 3 posts - 0 views - updated Apr 11 (5 days ago)

[ANN] Released OpenEngSB Framework 2.4.4

By Andreas Pieber - 1 post - 0 views - updated Apr 11 (5 days ago)

Proposal for 2.4.4

By Andreas Pieber - 7 posts - 1 view - updated Apr 11 (5 days ago)

Groups
5 of 1000+
Overview Discussion

OpenEngSB developer discussion

OpenEngSB Distribution Release Policy

7 posts by 3 authors in OpenEngSB developer discussion

Andreas Pieber

Apr 11 (5 days ago)

Sign in to reply

Hey guys,  
  
I've just released openengsb-framework 2.4.4; the release is interesting for ppl using the framework directly, so it was really worth the release; nevertheless it does not contain any ground shaking bug-fixes -> I'm curious if I should also do a openengsb-2.0.3 release. Any thoughts on this matter? Do we do a bundle release everytime we release any of it's components or only if there are direct results for the users of the bundle (or at least how we expect them to use it).  
  
Kind regards,  
Andreas

Christoph Gritschenberger

Apr 11 (5 days ago)

Sign in to reply

It would be quite a lot of releases, when we release a new bundle with every framework-release.  
Also wouldn't we have to release a new bundle on every domain/connector release?  
  
I think we should not waste time with too many releases here and rather provide some easy way to dynamically update components.  
  
kind regards,  
christoph

- show quoted text -

# Issue-Tracker

... Bug Tracker, Project Tracker

- Erfassung von Fehlern und Änderungswünschen
- Verwaltung von Roadmaps
- Feature-Beschreibungen und Arbeitspakete

# Integration von Issue-Tracker mit SCM

OpenEngSB / OPENENGSB-2848  
access to index in DefaultPersistenceIndex is not th

Comment Planning Board Task Board More Actions Reopen Issue Workflow

**Details**

Type: Bug  
Priority: Major  
Affects Version/s: None  
Component/s: framework  
Labels: None  
Similar Issues: None

Status: Closed  
Resolution: Fixed  
Fix Version/s: framework-2.4.3 ... (2)

**People**  
Assignee:  
Reporter:  
Vote

**Dates**  
Created:  
Updated:  
Resolved:

**Activity**

All Comments History Activity Commits

**Christoph Gritschenberger** 26/Mar/12 2:53 PM

[OPENENGSB-2848] synchronize access to index-array [View full commit](#)

master  
+13 -7 components/persistence/src/main/java/org/openengsb/core/persistence

**Christoph Gritschenberger** 26/Mar/12 2:53 PM

[OPENENGSB-2848] synchronize access to index-array [View full commit](#)

v2.4.x  
+13 -7 components/persistence/src/main/java/org/openengsb/core/persistence

**Christoph Gritschenberger** 26/Mar/12 2:53 PM

[OPENENGSB-2848] synchronize access to index-array [View full commit](#)

github [Signup and Pricing](#) [Explore GitHub](#)

openengsb / openengsb-framework

Code Network Pull Requests 3

Files Commits Branches 13

**[OPENENGSB-2848] synchronize access to index-array**

ChristophGr authored 21 days ago 1 parent fd7aad03b6 commit ff11c257f3b41bb8

Showing 1 changed file with 13 additions and 7 deletions.

components/persistence/src/main/java/org/openengsb/core/persistence/internal/DefaultPersistenceIndex.j

```

... @@ -63,13 +63,17 @@ public void indexObject(Class<?> bean, File beanLocation) {
63     objectInfo.addType(class1);
64
65     objectInfo.addType(bean);
66     index.add(objectInfo);
67     synchronized (index) {
68         index.add(objectInfo);
69     }
70     LOGGER.debug("Adding to index {}: {}", indexFile.toString(), objectInfo.toString());
71
72     @Override
73     public void removeIndexObject(ObjectInfo info) {
74         index.remove(info);
75         synchronized (index) {
76             index.remove(info);
77         }
78
79     @Override
80     public List<ObjectInfo> findIndexObject(Class<?> beanClass) {
81         LOGGER.trace("Looking for bean class {} in index {}", beanClass.getName(), indexFile.to
82         List<ObjectInfo> retVal = Lists.newArrayList();
83         for (ObjectInfo info : index) {
84             if (info.containsClass(beanClass)) {
85                 retVal.add(info);
86                 LOGGER.debug("Looking for {} in index {} and found " + info.toString(), beanClas
87                 indexFile.toString());
88             }
89         }
90     }

```

# Beispiele für Issue-tracker

- Frei
  - Bugzilla
  - Trac
  - Redmine
- Kommerziell
  - Atlassian JIRA
  - Microsoft Team Foundation Server
- Eigenentwicklungen
  - Github
  - Google Code

# Herausforderungen bei globaler Verteilung

- Große geographische Distanzen
  - Kein Face-2-Face
  - Austauschen von Handzeichnungen erschwert
- Verschiedene Arbeitszeiten
- Verschiedene Zeitzonen
- → Hauptsächlich asynchrone Kommunikation

# Synchrone Kommunikation

- Chat
  - IRC
  - XMPP
- VoIP/Video-Chat
  - Skype
  - XMPP (Google Talk)
- Cloud Document Services (Google Docs, Office 365)
- VNC (Skype, Teamviewer)

# Kulturen

- Linux
  - Eigene Mailinglisten für Bereiche
  - Durchreichen von Patches durch Hierarchie
  - „*Benevolent dictator governance model*“
- Apache
  - Project Management Comittee (PMC)
  - Entscheidungen durch „Votes“ über Mailinglisten
  - „*Meritocratic governance model*“
- OPS4J
  - Öffentlich schreibbare Github-repositories
  - Diskussionen auf Mailinglisten
  - „*Open Contribution*“
  - „*Wiki brought to Coding*“



# Zusammenfassung

- Synchrone vs. asynchrone Kommunikation
- Mailinglisten und Issue-Tracker
- Herausforderung bei globale Verteilung
- Verschiedene Projektmanagement-Varianten in OSS

# References

- Best Practice Software-Engineering, Eine praxiserprobte Zusammenstellung von komponentenorientierten Konzepten, Methoden und Werkzeugen - Alexander Schatten et al
- <http://martinfowler.com/articles/injection.html>
- <http://www.oss-watch.ac.uk/resources/benevolentdictatorgovernancemodel.xml>
- <http://www.oss-watch.ac.uk/resources/meritocraticGovernanceModel.xml>