



investdwin

Software-Product-Line Platform

Agenda

1. Introduction
2. Software-Product-Line
3. Development Environment
4. Configuration Management
5. Assets
6. Binding Variability
7. DDD and MDA
8. Conclusion

Feel free to ask questions!

1. Introduction

Story

Motivation

1. Introduction

- Background -

- Bachelor Thesis -> Companies platform vs SPL (Software-Product-Line)?
 - Company decided to stick to their business components
 - Blueprint for projects instead of SPL
- Decided to research SPL further as a side project
 - Domain of Algorithmic Trading, SPL as foundation
 - Research Project
 - Suitable for any Team Size
 - low/no running costs
 - OpenSource **Leverage**
 - **Reuse** has highest priority
 - R&D requires **Flexibility**
 - **Development Comfort** very important
 - It has to be **FUN!**



In development since October 2009

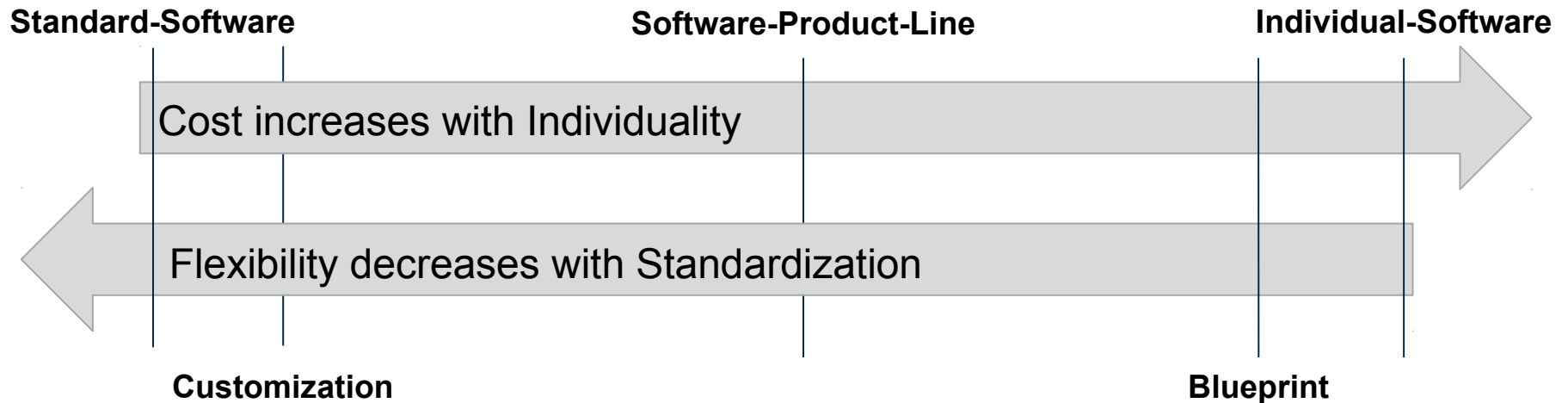
relatively stable since 2012

Open Source since 2016

2. Software-Product-Line

Scientific Background

2. Software-Product-Line - Classification -

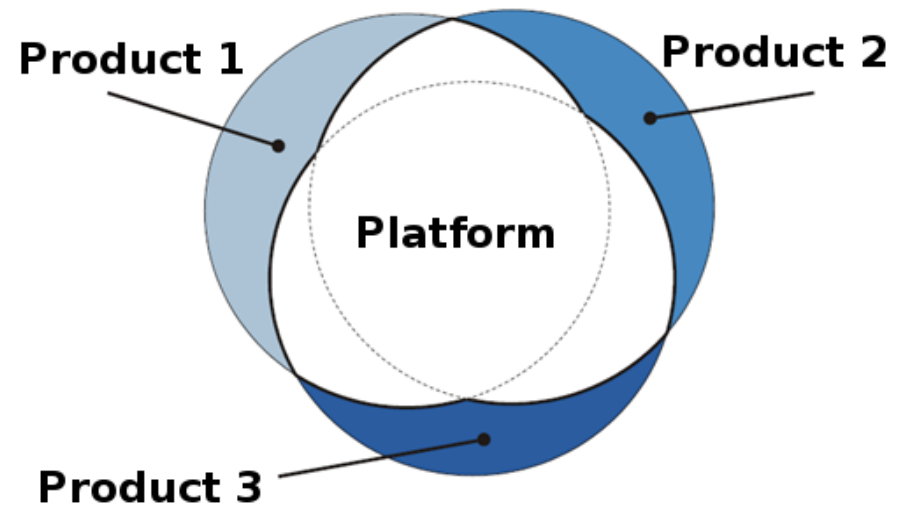


Solution:

- ✓ Reduce costs through **Reuse**
- ✓ Keep flexibility through **Variability**

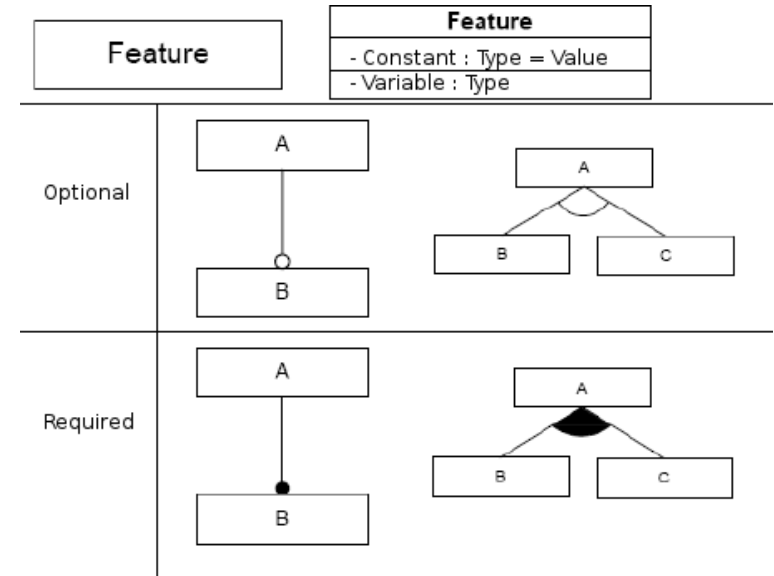
2. Software-Product-Line - Assets -

- Assets
 - Artifacts such as **Documentation, Code, Libs, Configuration, Products, Platform**
 - SPL is made of - and manages Assets
- Reuse
 - Artifacts are designed to be reused
 - Creation of **Synergies**
 - Reduction of Development Cost
 - Faster Time-To-Market
 - Avoid Copy/Paste (Artifact duplication)
- Modularity
 - **Bundle Artifacts**
 - Only deliver what the customer paid for!



2. Software-Product-Line - Variability -

- Variability
 - Interchangeability of used Implementation
 - Selection of **Features**
 - Make price variable and align it to the customer
- Variability Points
 - Decision „**where**“ Variants can be chosen
 - Planned Flexibility Options
 - Configurable or Chooseable
 - Customer makes the Decision here
- Variants
 - Encapsulate Features to be chosen optionally
 - Compare with Car Manufacturing:
 - Coupling Device yes/no
 - Sport Suspension vs Comfort Suspension
 - Combi-Van vs Limousine



The screenshot shows the Mercedes-Benz online configurator interface. It displays various car models (S-Class, E-Class, C-Class) and their corresponding prices. The interface includes a navigation bar with steps: 1. Modell wählen, 2. Motorisierung, 3. Designlinie, 4. Lack & Polster, 5. Ausstatt., 6. Ihr Fahrzeug, 7. Ihr Kontakt. The main content area shows the '5. Ausstattung' (Equipment) section, which lists various optional features and their prices. The right side of the interface shows the 'Ihre Wahl / Preis (EUR)' (Your choice / Price in EUR) section, which displays the selected car model (C 180 Kompressor Limousine) and its total price (38,732.28 EUR). The bottom of the interface shows the 'Fertig' (Finish) button and the 'Internet' icon.

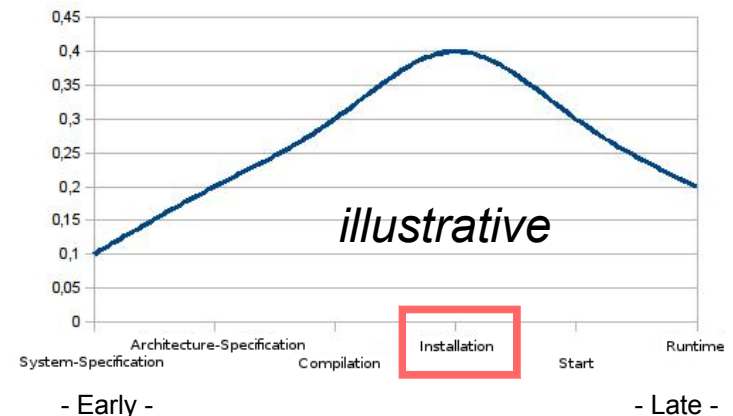
2. Software-Product-Lines - Binding -

- Binding
 - Variability Point gets placed with a Variant
- Binding Points
 - Decision „**how**“ in Code/Deployment a Variant gets chosen
 - Patterns: Dependencies, Properties, Runtime-Button
- Binding Times
 - Decision „**when**“ a Variant can be chosen
 - Compare with a finished Car:
 - Choose Equipment (when Buying)
 - Update Navigation-Maps (before Engine Start)
 - Activate Sportgear (while Driving)
 - Deactivate Electronic Stability Control (while Driving)

Overview of Binding Times:

	flexibility	performance	code size	complexity
source time	-	+	+	-
compile time	+	+	+	-
link time	+	+	+	-
load time	++	+	+	+
run time	+++	-	-	+

Usage of Binding Times:



3. Development Environment

What is the SPL realized in?

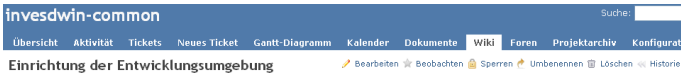
Standard Stuff...

3. Development Environment - Low Impact -

- **OS:** Windows/Linux
- **Versioning:** SVN/Git
- **Language:** Java 8 (started with Java 6)
- **IDE:** Eclipse + Plugins
- **Build & Dependency Mgmt:** Maven

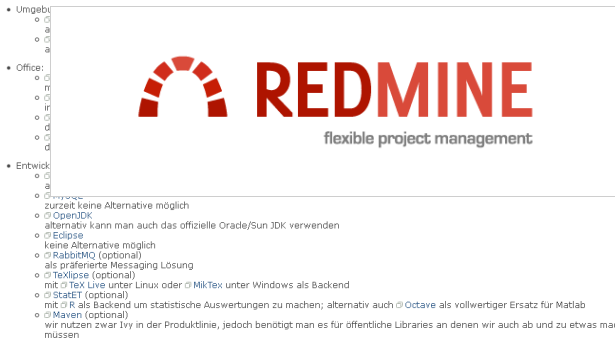
Earlier Ant+Ivy+Groovy was used (basis for this concept), later reimplemented with Maven to improve build times.

- **Documentation:**

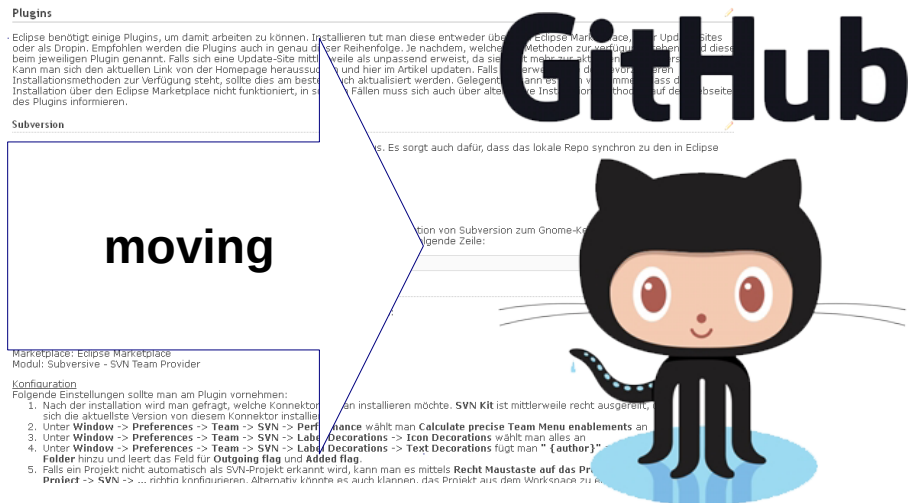
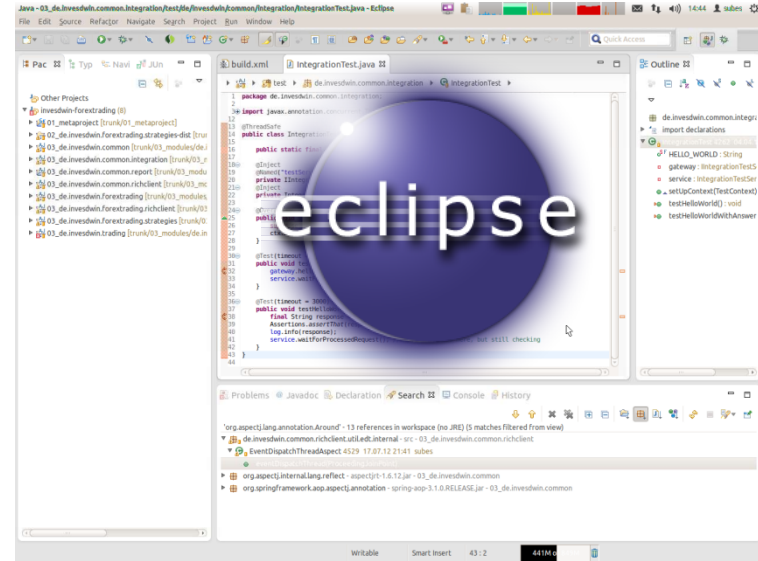


Wir nutzen immer sofern möglich die aktuellste Version der jeweils eingesetzten Software. Die wichtigsten Grundkomponenten sind Java und Eclipse. Wie ein System auf dieser Basis zu konfigurieren ist, wird in diesem Artikel beschrieben. Die Software ist in der Regel so ausgewählt, dass sie sowohl unter Windows als auch Linux als Umgebung zur Verfügung steht. Empfohlen wird jedoch Linux, daher beschreibt dieser Artikel wie man sich Ubuntu als empfohlene Umgebung einrichtet.

Aus folgenden Bestandteilen besteht die Entwicklungsumgebung:



WARNUNG: Bei dieser Softwareauswahl handelt es sich um Lizenztechnisch unbedenkliche Lösungen. Falls ein Entwickler etwas anderes unter eigener Lizenz verwenden möchte, tut er dies auf eigene Verantwortung und Gefahr.



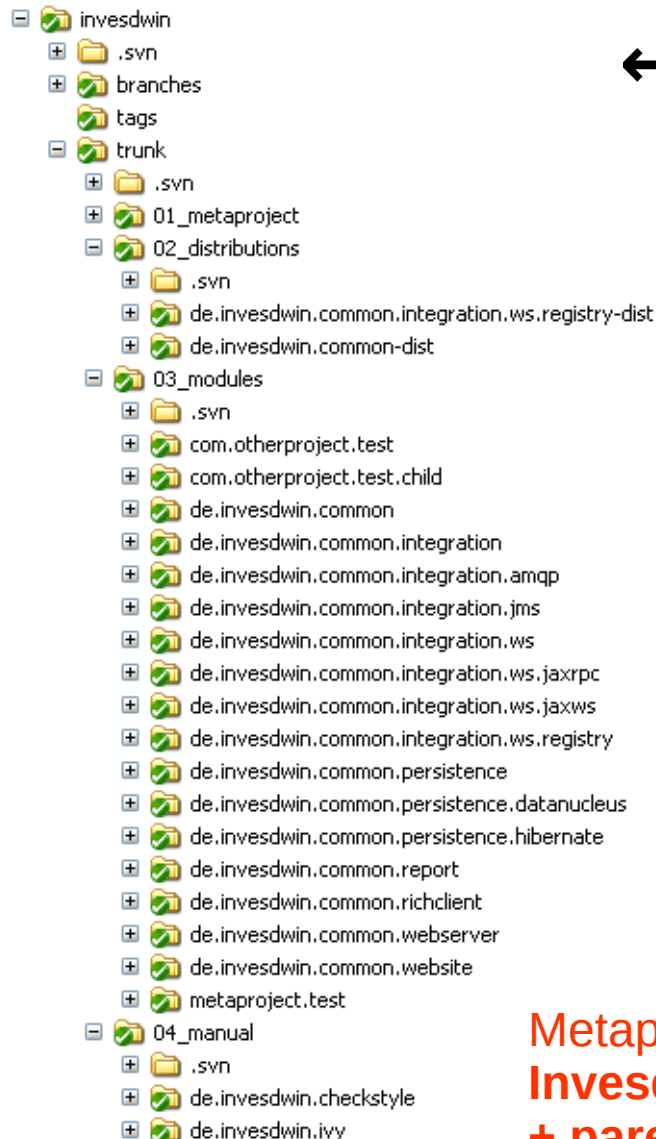
4. Configuration Management

New Approach

4. Configuration Management

- Overview -

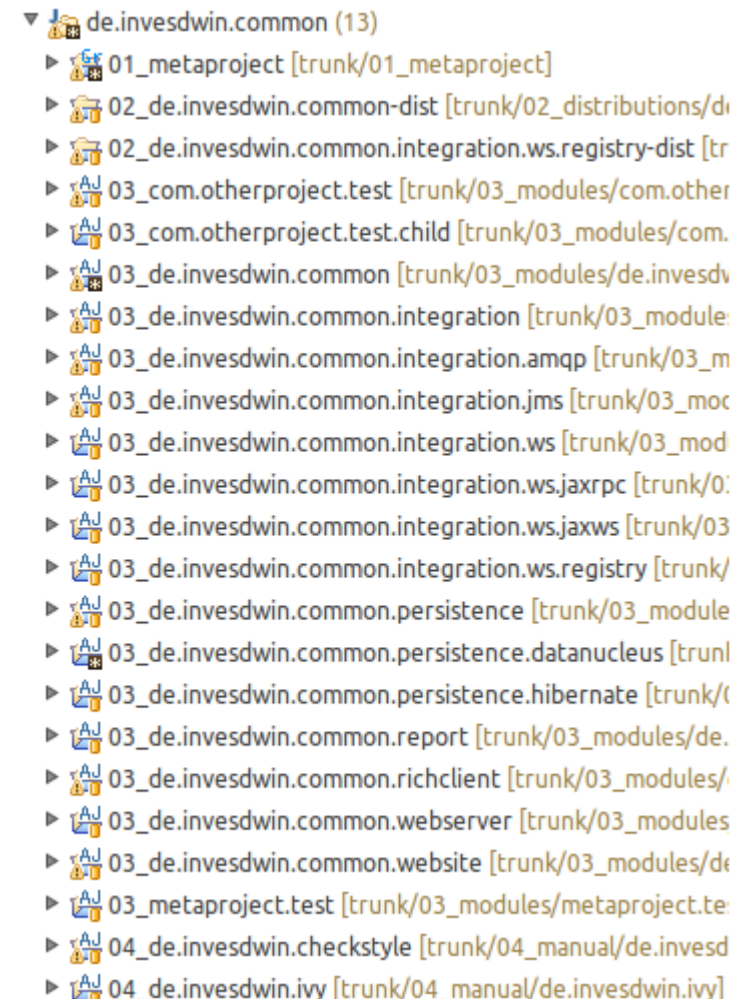
REPLACED BY MAVEN
with a better implementation of this concept



← **Explorer**
Eclipse →

Product:

- 01 Metaproject
- 02 Distribution
- 03 Module
- (04 manual Module)



Metaproject got turned into:
Invesdwin-maven-plugin
+ parent pom.xmls

4. Configuration Management

- Module I -

- **Reusable Building Blocks for Products**

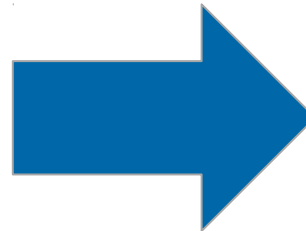
- **Technology:** Best Practices, Patterns, Frameworks, Utils, Tools
- **Domain:** Services, Entities, Logic, Algorithm

→ Each has its own Eclipse-Project to ensure Modularity

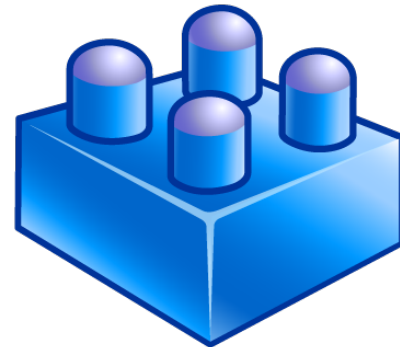


Components

Insurer, Contract, Customer, ...



Combine &
make Reusable



Module

Contract Management

4. Configuration Management

- Module II -

- Comparison to previous Platform:

- Module > Container**

- Upgrade-Path** instead of Copy-Paste

- Rather add Variability to a Module, instead of creating another one

- Goal: Effectively less Maintenance Cost

Previous Platform had a monolithic architecture where services/beans/components were bundled into „containers“ that had tight coupling among each other.

A module in this sense can bundle multiple containers, thus is more than a container.

Fix Bug multiple times differently

VS

Fix Bug once + update Version multiple times

- Apply Lessons Learned from old Business Components:

- YAGNI & KISS**

- Do not develop on **Green Field** and avoid **Over-Engineering**

- Only add Variability when it is actually needed

- Enforce Loose Coupling**

- Utilize submodules to encapsulate Functionality/Alternatives

4. Configuration Management

- Module III -

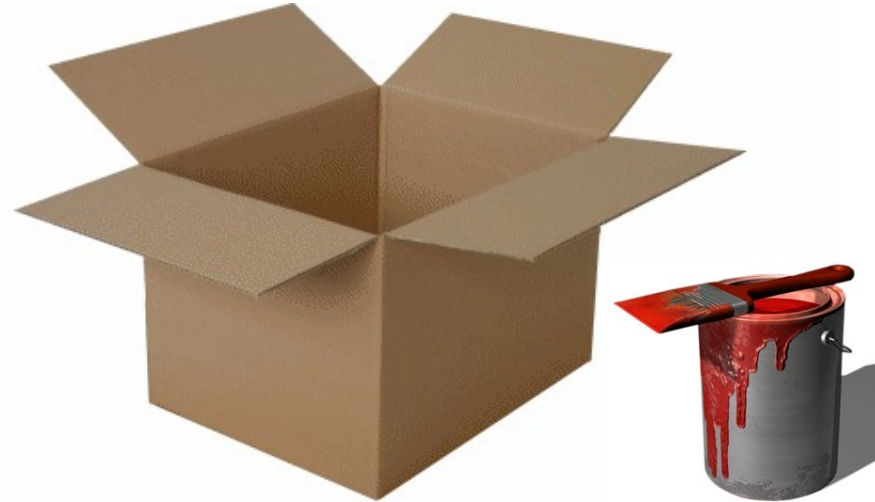
- **Why no OSGi?**
 - Dynamic Load/Unload not needed
 - OSGi-Descriptors not maintained well by Open Source Projects
 - Avoid Classpath-Problems (one common Classpath is easier)
 - Jar-Hell and Version-Conflicts already solved by Dependency Management
 - Loose Coupling via „internal“ Packages enforced by Checkstyle-Rule
 - reasoned deviations from the „internal“ Package Rule possible
 - without technical challenges



**only
OSGi
Compatible**

4. Configuration Management - Distribution -

- **A configurable, deployable Product**
 - Bundles multiple Modules
 - **Configuration of Variability** via
 - Dependencies
 - Properties
 - Additional Resources
 - Customer Specific
 - Target Environment Specific
- Decision about package type only here:
 - Executable **Fat-Jar**
 - **Zip** with launch scripts
 - **War** as Container

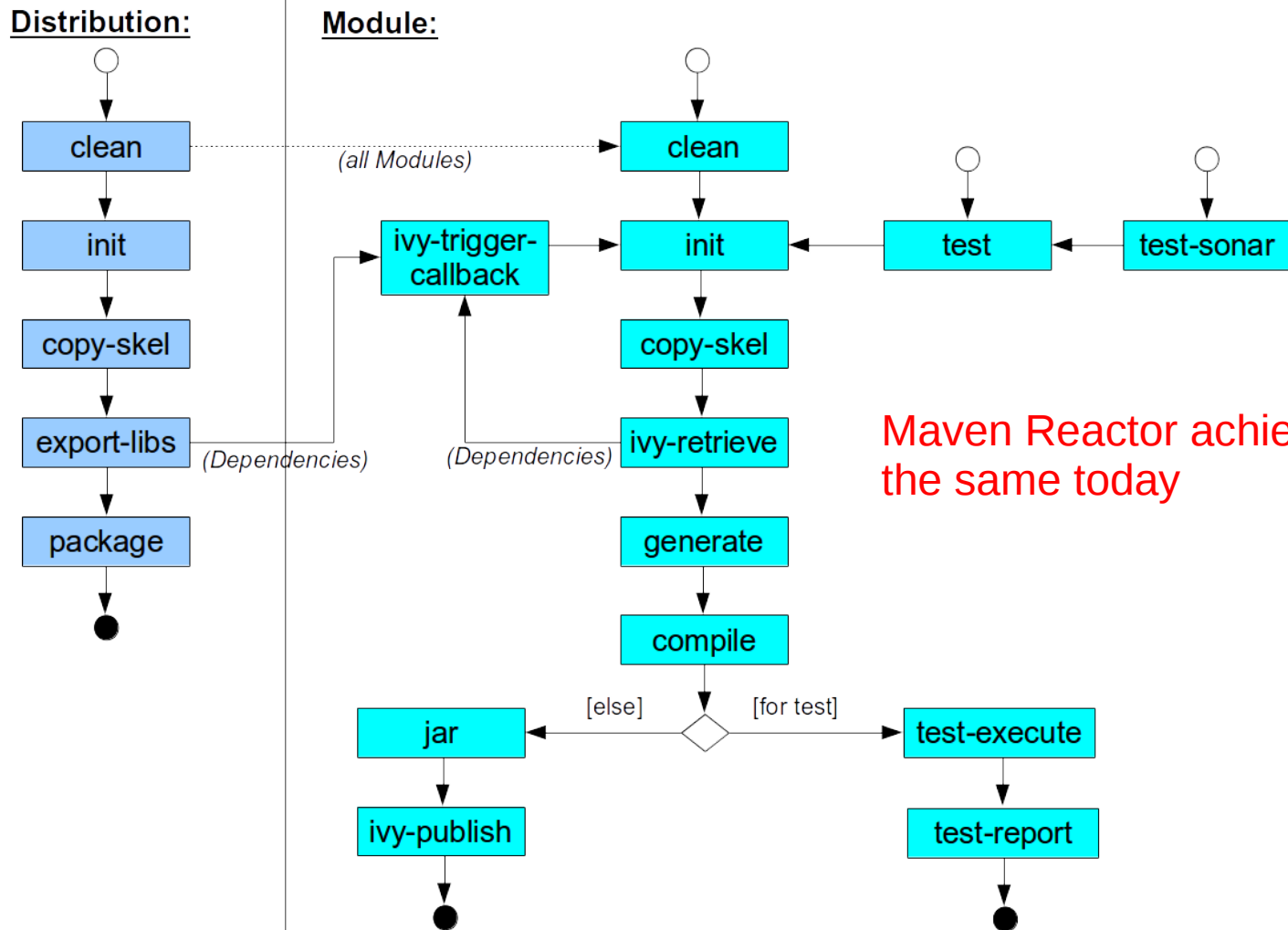


→ Example Web-Application: Fat-Jar with embedded Jetty **or** War for Tomcat?

4. Configuration Management - Metaproject I -

REPLACED BY MAVEN
with a better implementation of this concept

▪ Definition of unified Build-Process:

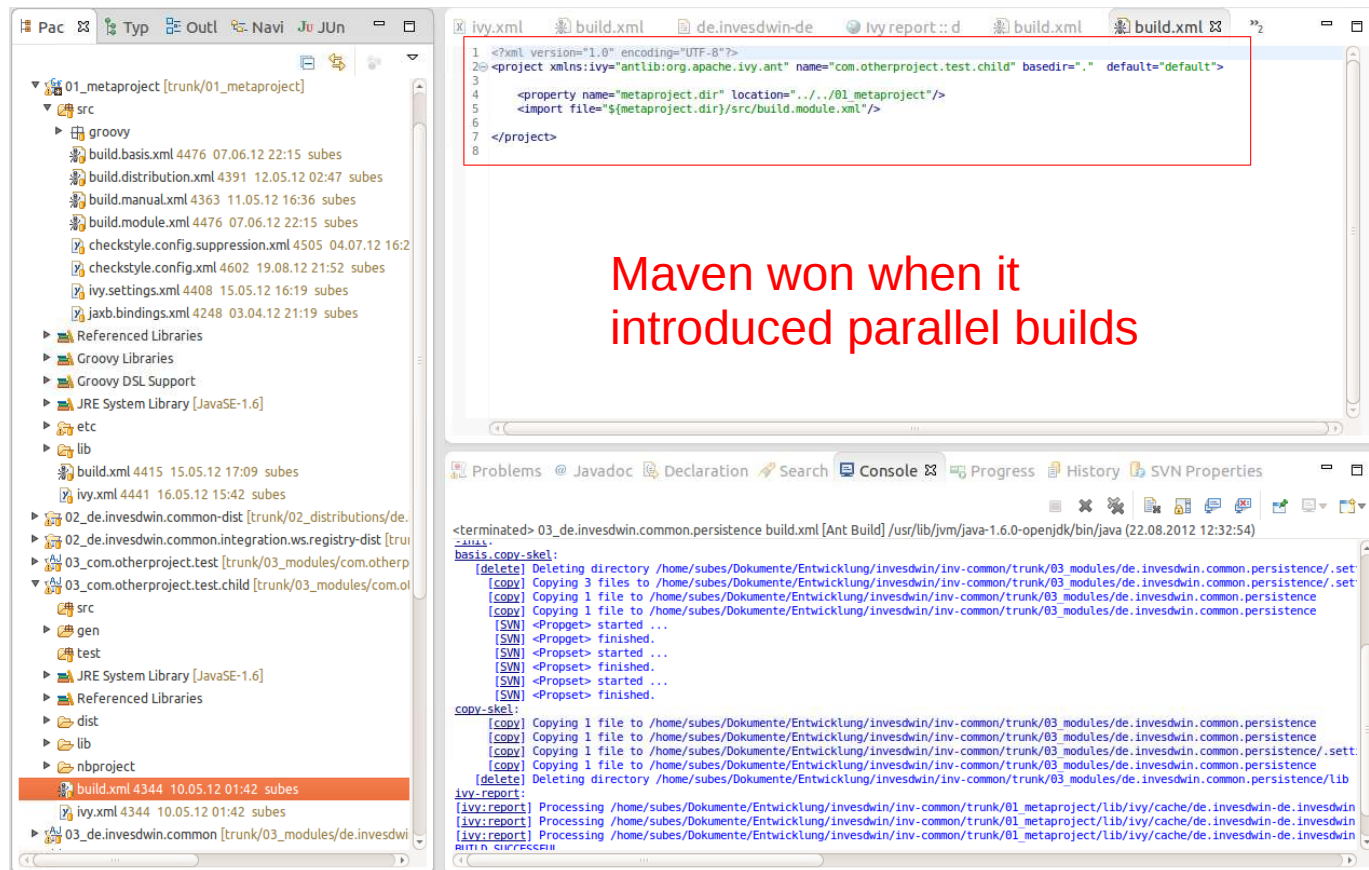


Maven Reactor achieves
the same today

4. Configuration Management - Metaproject II -

REPLACED BY MAVEN
with a better implementation of this concept

- **Reusable standardized Ant-Scripts** → maintain in only one place
- **Project Templates for Modules and Distributions** → à la Maven Archetype
- **Other Assets inherit Eclipse-Project-Configuration** → configure only once



4. Configuration Management - Metaproject III -

REPLACED BY MAVEN
with a better implementation of this concept

■ Automatically generated Eclipse-Classpath for Dependencies:

- Project-References for multiple modules spanning Refactorings
- Unified Naming of Dependencies (<Module>-<Version>.jar)
- Sources for all Jars automatically linked
- AspectJ and other Eclipse-Plugins automatically configured

M2E Plugin for Eclipse
achieves the same today

→ Reduction of maintenance effort for Modules, only configuration in **ivy.xml** required!

```
Dashboard build.module.xml ivy.xml build.xml de.invesdwin-de.invesdwin.common.per ivy report :: c
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <ivy-module version="2.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:noNamespaceSchemaLocation="http://ant.apache.org/ivy/schemas/ivy.xsd"
4   xmlns:mv="http://ant.apache.org/ivy/maven">
5   <info
6     organisation="de.invesdwin"
7     module="${ant.project.name}"
8     status="release"
9     revision="${ivy.version.invesdwin}"
10  </info>
11  <configurations>
12    <conf name="default"/>
13    <conf name="tests" />
14    <conf name="sources" />
15  </configurations>
16  <publications defaultconf="default">
17    <artifact conf="default" type="jar" ext="jar"/>
18    <artifact conf="tests" type="jar" ext="jar" classifier="tests"/>
19    <artifact conf="sources" type="source" ext="jar" classifier="sources"/>
20  </publications>
21  <dependencies>
22    <!-- spring orm -->
23    <dependency org="org.springframework" name="spring-orm" rev="${ivy.version.spring}" conf="default->master;sources" />
24    <dependency org="org.springframework" name="spring-jdbc" rev="${ivy.version.spring}" conf="default->master;sources" />
25    <dependency org="com.mysema.querydsl" name="querydsl-jpa" rev="${ivy.version.querydsl}" conf="default->master;sources"/>
26
27    <!-- connection pooling -->
28    <dependency org="com.jolbox" name="bonecp" rev="${ivy.version.bonecp}" conf="default->master;sources" />
29
30    <!-- jpa metamodel generator -->
31    <dependency org="org.hibernate" name="hibernate-jpamodelgen" rev="${ivy.version.hibernate.jpamodelgen}" conf="default->master;sources" />
32
33    <!-- Test Konnektoren für Datenbanken -->
34    <dependency org="com.h2database" name="h2" rev="${ivy.version.h2}" conf="tests->master;sources" />
35    <dependency org="mysql" name="mysql-connector-java" rev="${ivy.version.mysql-connector}" conf="tests->master;sources" />
36    <dependency org="de.invesdwin" name="de.invesdwin.common" rev="${ivy.version.invesdwin}" conf="default;tests;sources" />
37  </dependencies>
38  </ivy-module>
```

Dashboard build.module.xml ivy.xml build.xml de.invesdwin-de.invesdwin.common.per ivy report :: de.invesdwin.common.pe

file:///home/subes/Dokumente/Entwicklung/invesdwin/inv-common/trunk/03_modules/de.invesdwin.common.persistence/lib/1/de.invesdwin-de.invesdwin.common.persiste

de.invesdwin.common.persistence 0.2.0 by de.invesdwin

resolved on 2012-08-22 12:32:58

default tests sources

Dependencies Stats

Modules	59
Revisions	59 (0 searched, 0 downloaded, 0 evicted, 0 errors)
Artifacts	59 (0 downloaded, 0 failed)
Artifacts size	14252 kB (0 kB downloaded, 14252 kB in cache)

Dependencies Overview

Module	Revision	Status	Resolver	Default	Licenses	Size
de.invesdwin.common by de.invesdwin	0.2.0	release	temp	false		244 kB
hibernate-jpamodelgen by org.hibernate	1.1.1.Final	release	invesdwin	false	Apache License, Version 2.0	157 kB
bonecp by com.jolbox	0.7.2-SNAPSHOT	Integration	repo	false		116 kB
querydsl-jpa by com.mysema.querydsl	2.3.0	release	invesdwin	false		94 kB
spring-orm by org.springframework	3.2.0.M1	release	invesdwin	false	The Apache Software License, Version 2.0	374 kB
spring-jdbc by org.springframework	3.2.0.M1	release	invesdwin	false	The Apache Software License, Version 2.0	394 kB

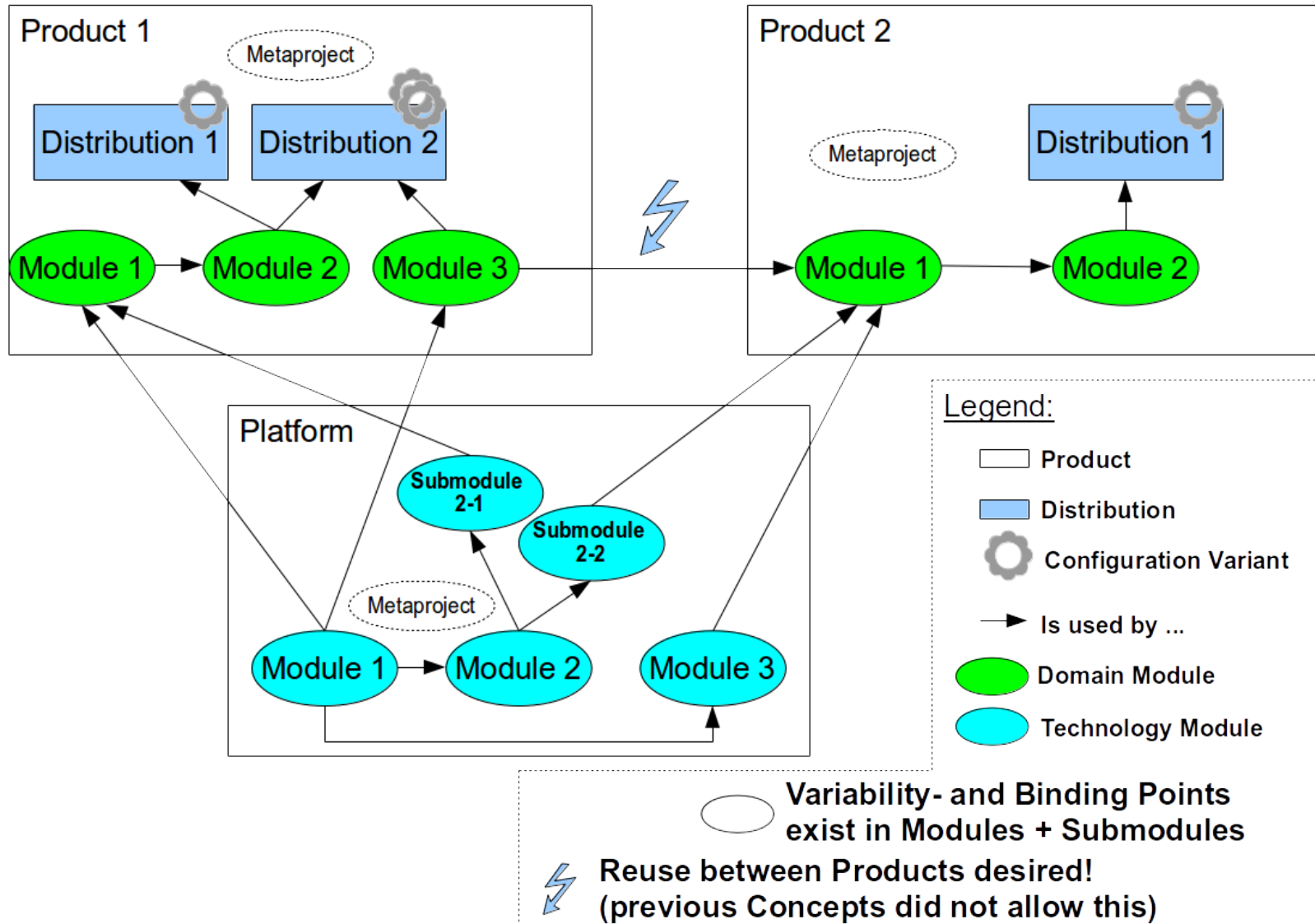
Details

de.invesdwin.common by de.invesdwin

Revision: 0.2.0

status	release
publication	20120809132657
resolver	temp
configurations	default

4. Configuration Management - Multi-Product-Management -



5. Assets

What exists already?

Platform, Products, Components

5. Assets - Platform I -

Outdated Information

some components/names have changed
also today there are more modules

metaproject



investdwin-common



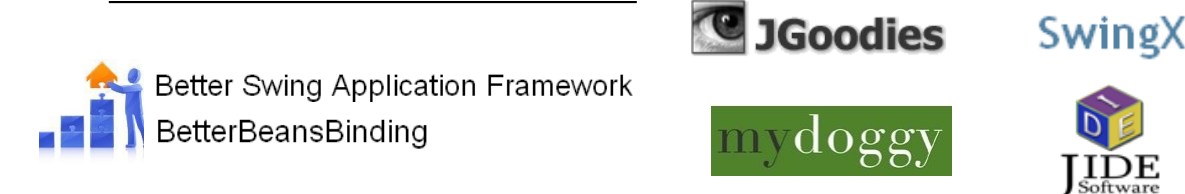
investdwin-common-persistence



investdwin-common-report



investdwin-common-richclient



Infrastructure



Dependency Repos



Google WindowBuilder Pro



5. Assets

- Platform II -

Outdated Information

some components/names have changed
also today there are more modules

investdwin-common-integration

- investdwin-common-integration-amqp



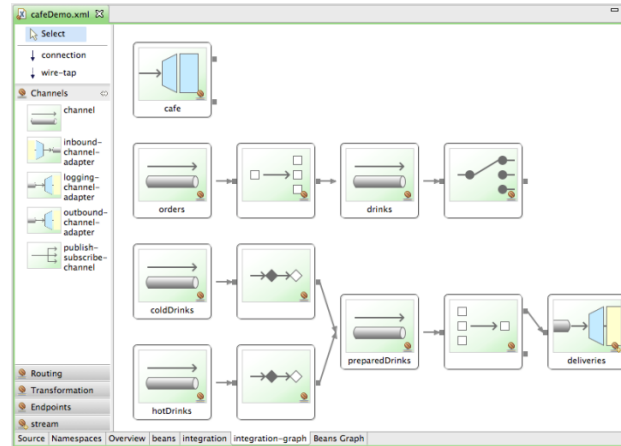
- investdwin-common-integration-jms



- investdwin-common-integration-ws



SPRING INTEGRATION



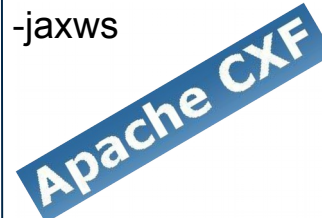
<?xml?>



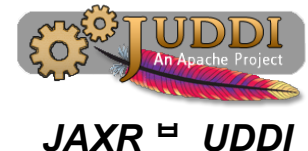
-jaxrpc



-jaxws



-registry



investdwin-common-website



investdwin-common-webserver

Powered by



Deployment in Tomcat as WAR
configurable in Distribution



5. Assets

- Example Product webproxy -

- invesdwin-webproxy(-...)



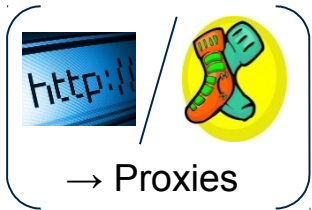
HtmlUnit

GUI-Less browser for Java programs



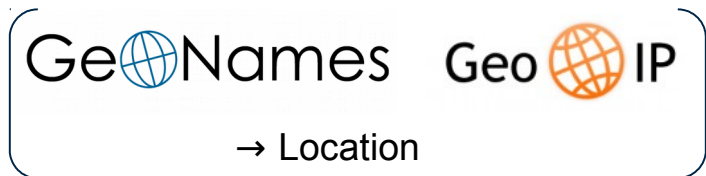
JAKARTA COMMONS

HTTPCLIENT



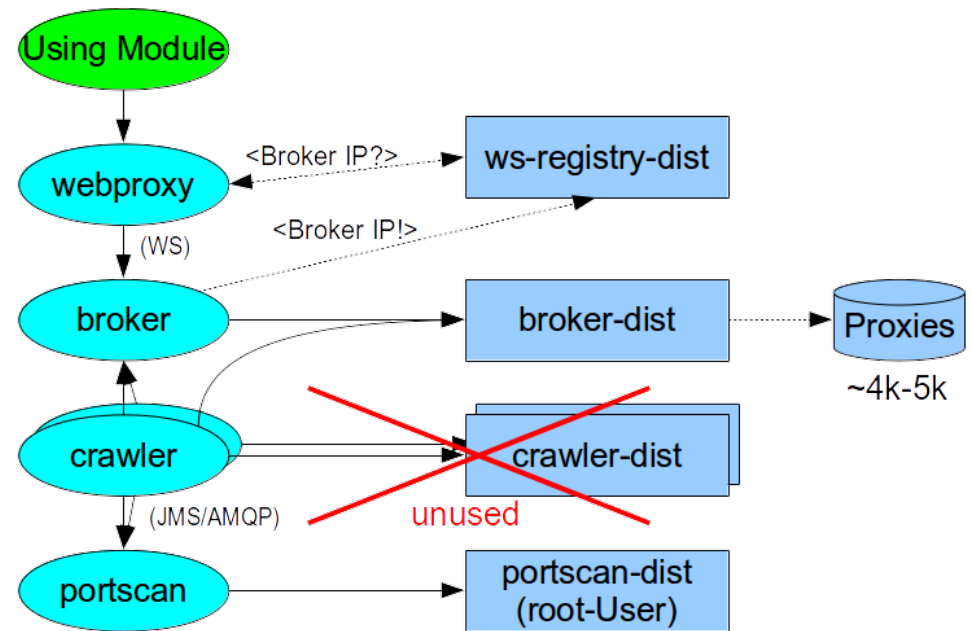
→ Proxies

win/lib/j - *Peap*



→ Location

- Simplified Context Diagram:



Legend:

Domain Module

Is used by ...

Technology Module

Distribution

Database

6. Binding Variability

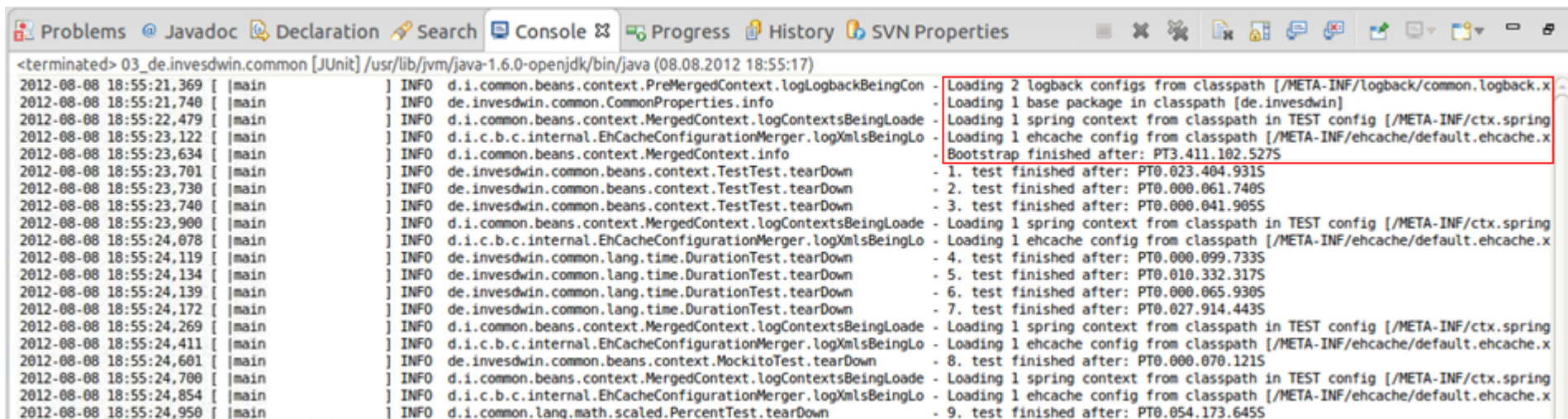
Bootstrap creates Flexibility

Pattern-Examples

6. Binding Variability

- Application Bootstrap -

- Configuration spread among various Modules
- Common Classpath (no OSGI)
- Two Spring ApplicationContexts
 - Premerged → collect and configure
 - Merged → running Application
- Per JUnit Test-Class a new Bootstrap (fast) for Configuration Changes:
 - Selecting Spring-XMLs via IContextLocation
 - In Test via setUpContext()
 - In Mocks/Stubs via IStub



```
<terminated> 03_de.invesdwin.common [JUnit] /usr/lib/jvm/java-1.6.0-openjdk/bin/java (08.08.2012 18:55:17)
2012-08-08 18:55:21,369 [main] INFO d.i.common.beans.context.PreMergedContext.logLogbackBeingCon
2012-08-08 18:55:21,740 [main] INFO de.invesdwin.common.CommonProperties.info
2012-08-08 18:55:22,479 [main] INFO d.i.common.beans.context.MergedContext.logContextsBeingLoade
2012-08-08 18:55:23,122 [main] INFO d.i.c.b.c.internal.EhCacheConfigurationMerger.logXmIsBeingLo
2012-08-08 18:55:23,634 [main] INFO d.i.common.beans.context.MergedContext.info
2012-08-08 18:55:23,701 [main] INFO de.invesdwin.common.beans.context.TestTest.tearDown
2012-08-08 18:55:23,730 [main] INFO de.invesdwin.common.beans.context.TestTest.tearDown
2012-08-08 18:55:23,740 [main] INFO de.invesdwin.common.beans.context.TestTest.tearDown
2012-08-08 18:55:23,900 [main] INFO d.i.common.beans.context.MergedContext.logContextsBeingLoade
2012-08-08 18:55:24,078 [main] INFO d.i.c.b.c.internal.EhCacheConfigurationMerger.logXmIsBeingLo
2012-08-08 18:55:24,119 [main] INFO de.invesdwin.common.lang.time.DurationTest.tearDown
2012-08-08 18:55:24,134 [main] INFO de.invesdwin.common.lang.time.DurationTest.tearDown
2012-08-08 18:55:24,139 [main] INFO de.invesdwin.common.lang.time.DurationTest.tearDown
2012-08-08 18:55:24,172 [main] INFO de.invesdwin.common.lang.time.DurationTest.tearDown
2012-08-08 18:55:24,269 [main] INFO d.i.common.beans.context.MergedContext.logContextsBeingLoade
2012-08-08 18:55:24,411 [main] INFO d.i.c.b.c.internal.EhCacheConfigurationMerger.logXmIsBeingLo
2012-08-08 18:55:24,601 [main] INFO de.invesdwin.common.beans.context.MockitoTest.tearDown
2012-08-08 18:55:24,700 [main] INFO d.i.common.beans.context.MergedContext.logContextsBeingLoade
2012-08-08 18:55:24,854 [main] INFO d.i.c.b.c.internal.EhCacheConfigurationMerger.logXmIsBeingLo
2012-08-08 18:55:24,950 [main] INFO d.i.common.lang.math.scaled.PercentTest.tearDown
- Loading 2 logback configs from classpath [/META-INF/logback/common.logback.x
- Loading 1 base package in classpath [de.invesdwin]
- Loading 1 spring context from classpath in TEST config [/META-INF/ctx.spring
- Loading 1 ehcache config from classpath [/META-INF/ehcache/default.ehcache.x
- Bootstrap finished after: PT3.411.102.527S
- 1. test finished after: PT0.023.404.931S
- 2. test finished after: PT0.000.061.740S
- 3. test finished after: PT0.000.041.905S
- Loading 1 spring context from classpath in TEST config [/META-INF/ctx.spring
- Loading 1 ehcache config from classpath [/META-INF/ehcache/default.ehcache.x
- 4. test finished after: PT0.000.099.733S
- 5. test finished after: PT0.010.332.317S
- 6. test finished after: PT0.000.065.930S
- 7. test finished after: PT0.027.914.443S
- Loading 1 spring context from classpath in TEST config [/META-INF/ctx.spring
- Loading 1 ehcache config from classpath [/META-INF/ehcache/default.ehcache.x
- 8. test finished after: PT0.000.070.121S
- Loading 1 spring context from classpath in TEST config [/META-INF/ctx.spring
- Loading 1 ehcache config from classpath [/META-INF/ehcache/default.ehcache.x
- 9. test finished after: PT0.054.173.645S
```

6. Binding Variability

- Properties I -

- All Properties are System-Properties
- Thus available in:
 - other Properties Files à la Ant \${property}
 - XML (Spring, Frameworks with Commons-Configuration)
 - Java (System.getProperty(„property“))
 - VisualVM (Monitoring, be aware of security!)

```
de.invesdwin.webproxy.portscan.properties
1 #On the host port 80 must be open and a service has to be running on it. The host also has to answer pings so that the check
2 de.invesdwin.webproxy.portscan.internal.PortscanProperties.CHECK_HOST=google.de
3 de.invesdwin.webproxy.portscan.internal.PortscanProperties.LOCAL_BIND_PORT=44125
4 de.invesdwin.webproxy.portscan.internal.PortscanProperties.ICMP_RESPONSE_TIMEOUT=3 SECONDS
5 #For timings see: http://www.networkuptime.com/nmap/page09-09.shtml
6 de.invesdwin.webproxy.portscan.internal.PortscanProperties.UPLOAD_PAUSE_BETWEEN_PACKETS=0 MILLISECOND
7 de.invesdwin.webproxy.portscan.internal.PortscanProperties.UPLOAD_PAUSE_BETWEEN_PACKETS_PER_HOST=0 MILLISECOND
8 de.invesdwin.webproxy.portscan.internal.PortscanProperties.RESPONSE_TIMEOUT_BETWEEN_SYN_PACKETS_PER_HOST=500 MILLISECOND
9 de.invesdwin.webproxy.portscan.internal.PortscanProperties.MAX_OPEN_TCP_REQUESTS=25
10 de.invesdwin.webproxy.portscan.internal.PortscanProperties.MAX_OPEN_SYN_REQUESTS=10
```

```
ctx.persistence.test.memory.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xmlns:context="http://www.springframework.org/schema/context"
4     xsi:schemaLocation="http://www.springframework.org/schema/beans
5         http://www.springframework.org/schema/beans/spring-beans.xsd">
6
7     <bean id="persistenceProperties" class="de.invesdwin.common.system.properties.SystemPropertiesDefinition">
8         <property name="systemProperties">
9             <map>
10                 <entry key="javax.persistence.jdbc.driver" value="org.h2.Driver" />
11                 <entry key="javax.persistence.jdbc.url" value="jdbc:h2:mem:invesdwin;DB_CLOSE_ON_EXIT=FALSE" />
12                 <entry key="javax.persistence.jdbc.user" value="sa" />
13                 <entry key="javax.persistence.jdbc.password" value="sa" />
14                 <entry key="hibernate.dialect" value="org.hibernate.dialect.H2Dialect" />
15                 <entry key="hibernate.hbm2ddl.auto" value="create" />
16             </map>
17         </property>
18     </bean>
19
20     <import resource="actx.persistence.hibernate.xml" />
21
22 </beans>
```

```
*PortscanProperties.java
1 package de.invesdwin.webproxy.portscan.internal;
2
3 import java.net.InetAddress;
4
5 @Immutable
6 public final class PortscanProperties {
7
8     public static final InetAddress CHECK_HOST;
9     public static final int CHECK_PORT = 80;
10    public static final int LOCAL_BIND_PORT;
11    public static final Duration ICMP_RESPONSE_TIMEOUT;
12
13    private static final SystemProperties SYSTEM_PROPERTIES = new SystemProperties(PortscanProperties.class);
14
15    static {
16        CHECK_HOST = NetworkUtil.toAddress(SYSTEM_PROPERTIES.getString("CHECK_HOST"));
17        LOCAL_BIND_PORT = readLocalBindPort();
18        ICMP_RESPONSE_TIMEOUT = SYSTEM_PROPERTIES.getDuration("ICMP_RESPONSE_TIMEOUT");
19    }
20
21    private PortscanProperties() {}
22
23    private static int readLocalBindPort() {
24        final String key = "LOCAL_BIND_PORT";
25        final Integer value = SYSTEM_PROPERTIES.getInt(key);
26        Assertions.assertThat(NetworkUtil.isPort(value))
27            .as(SYSTEM_PROPERTIES.getErrorMessage(key, value, null, "Value must be inclusively between "
28                + NetworkUtil.PORT_MIN + " and " + NetworkUtil.PORT_MAX + "."))
29            .isTrue();
30        return value;
31    }
32 }
```

Applications

- Local
 - VisualVM
 - de.invesdwin.common.integration.w
 - de.invesdwin.financialdata.crawler
 - de.invesdwin.webproxy.broker-prod
 - Remote
 - VM CoreDumps
 - Snapshots

Start Page x de.invesdwin.webproxy.broker-prod-0.2.0.jar (pid 15284) x

de.invesdwin.webproxy.broker-prod-0.2.0.jar (pid 15284)

Overview

PID: 15284
Host: localhost
Main class: de.invesdwin.webproxy.broker-prod-0.2.0.jar
Arguments: <none>

JVM: OpenJDK 64-Bit Server VM (22.0-b10, mixed mode)
Java: version 1.7.0_03, vendor Oracle Corporation
Java Home: /usr/lib/jvm/java-7-openjdk-amd64/jre
JVM Flags: <none>

Heap dump on OOME: disabled

JVM arguments	System properties
de.invesdwin.webproxy.WebproxyProperties.PROXY_POOL_COOLDOWN_MIN_TIMEOUT=100 MILLISECOND	de.invesdwin.webproxy.WebproxyProperties.PROXY_POOL_WARMUP_TIMEOUT=10 MINUTES
de.invesdwin.webproxy.WebproxyProperties.PROXY_VERIFICATION_REDIRCT_SLEEP=15 SECONDS	de.invesdwin.webproxy.WebproxyProperties.PROXY_VERIFICATION_RETRY_ON_ALL_EXCEPTIONS=false
de.invesdwin.webproxy.broker.internal.BrokerProperties.ADDITIONAL_RANDOM_TO_BE_SCANNED_PORTS_PERCENT=25	de.invesdwin.webproxy.broker.internal.BrokerProperties.MAX_SPECIFIC_TO_BE_SCANNED_PORTS=1000
de.invesdwin.webproxy.broker.internal.BrokerProperties.PROXY_DOWNTIME_TOLERANCE=18 HOURS	de.invesdwin.webproxy.crawler.internal.CrawlerProperties.RANDOM_SCAN_ALLOWED=false
de.invesdwin.webproxy.crawler.internal.CrawlerProperties.WAIT_FOR_PORTSCAN_PROCESSING_END=true	de.invesdwin.webproxy.geolocation.internal.GeolocationProperties.GEOIP_DATA_URL=http://geolite.maxmind.com/download/geoi
de.invesdwin.webproxy.geolocation.internal.GeolocationProperties.GEONAMES_DATA_URL=http://download.geonames.org/expo	ehcache.disk.store.dir=/tmp/15284@invesdwin.de/ehcache
ehcache.disk.store.dir=/tmp/15284@invesdwin.de/ehcache	ehcache.disk.store.dir=/tmp/15284@invesdwin.de/ehcache
file.encoding=UTF-8	file.encoding.pkg=sun.io
file.separator=/	hibernate.dialect=org.hibernate.dialect.MySQLInnoDBDialect
hibernate.hbm2ddl.auto=update	hibernate.hbm2ddl.auto=update
java.awt.graphicsenv=sun.awt.X11GraphicsEnvironment	java.awt.printerjob=sun.print.PSPrinterJob

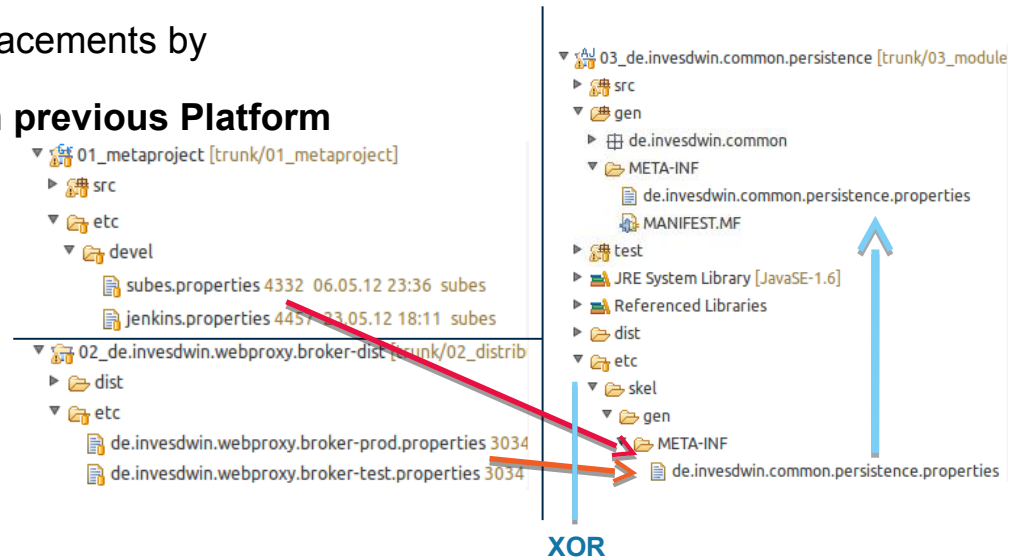
6. Binding Variability - Properties II -

Now per Module Overrides
with Maven Implementation

- **Default-Values** in Module-Properties and Replacements by

Developer- and **Distribution-Properties** as in previous Platform

- Build-Process **replaces** the Properties in
references Module-Projects (in **gen-Source**)
and retrieved invesdwin-Libs (in **Jar**)



- **Variable Binding-Time** with the following Priority for Overrides:

1. Source-Time: Default-Values in Properties-File
or Spring-XML
2. Build-Time: Developer- or Distribution-Properties
3. Load-Time: Spring-XML
or Java-Parameter via `-Dproperty=value`
4. Runtime: Java via `Properties.setProperty(„value“)`

	flexibility	performance	code size	complexity
source time	-	+	+	-
compile time	+	+	+	-
link time	+	+	+	-
load time	+++	+	+	+
run time	+++	-	-	+

6. Binding Variability - Contract Modules -

- **Book:** Enterprise Integration Patterns

- **Framework:** Spring Integration (SI)

- **SPL-Usage:**

- Separate Interface and Implementation
- Enforce loose coupling
- Technological indifference to Communication-Type

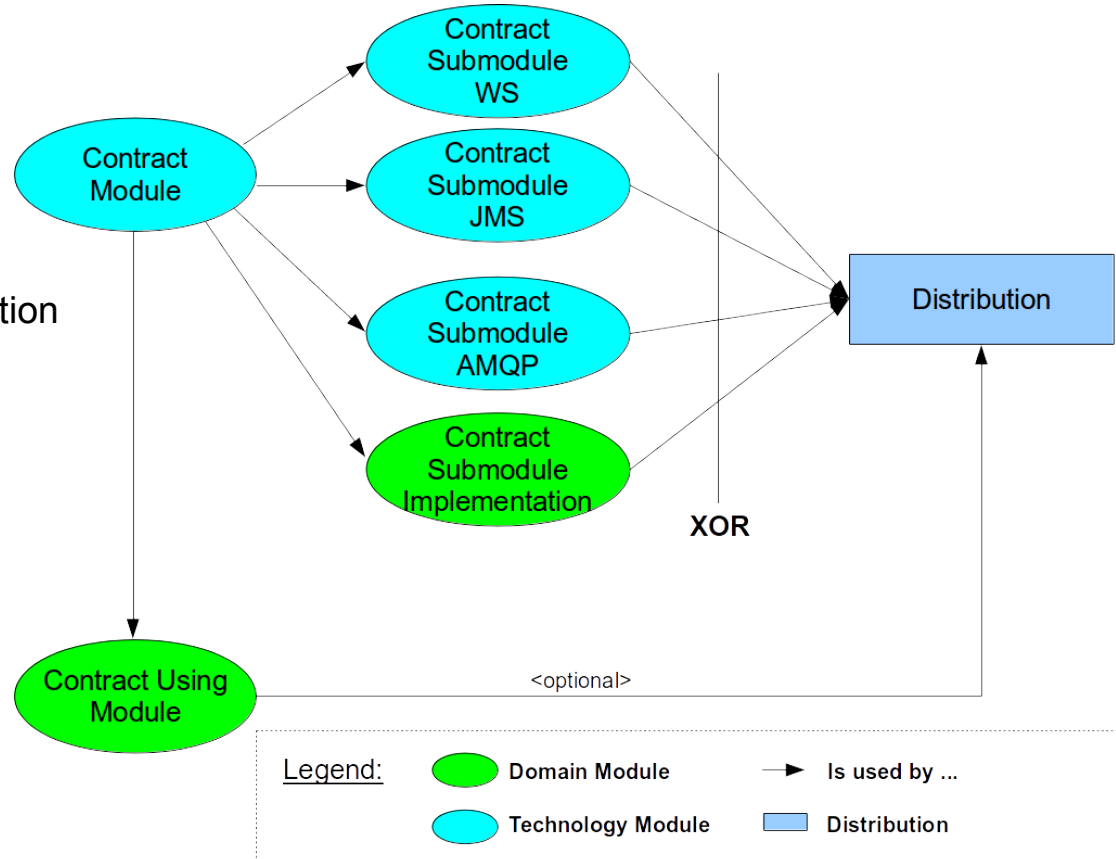
- **Contract:** Interfaces

- **Contract Submodule:** SI-Configuration

- **Contract Impl:** Interface-Implementation

- **Possibilities:**

- Communication-Overhead reduced by putting Implementation in Distribution directly
- Multiple Instances of Impl with Fail-Over or Load-Balancing



7. DDD and MDA

DDD!

Where could MDA be used?

7. DDD und MDA

- Concepts -

- **DDD – Domain Driven Design/Development**

- Technology Modules in English
- Domain Modules in Domain-Language
- Frontend internationalized → no Module copies that get translated

- **MDA – Model Driven Architecture**

- As of now no MDA:
 - Prefer Framework over Generator → easier Maintenance, simpler Build-Process
 - Research Project:
 - Less recurring Concepts
 - Initial invest for Generator would not provide ROI fast enough
- MDA possible:
 - Code-Generation at Module-Level, not all-encompassing
 - One Model per Module
 - Extension of „generate“-Build-Step
 - Lots of SPL-Productivity-Potential



8. Conclusion

Benefits

Drawbacks

8. Conclusion

- Your opinion is very welcome! -

Benefits	Drawbacks
+ SPL not just theory, but usable	- Learning Curve
+ High Development Comfort	- New Platform
+ High Flexibility	- New Technologies
+ Cheaper Maintenance for complex Projects and in Multi-Project-Environment	- Change of Organizational Structure needed?
+ Matches most Features/Requirements of previous platform	- More Diversity to manage
+ Standardized Eclipse-Projects and Build-Scripts	
+ Open, widely used technologies	
+ Reuse as highest goal	



Thanks a lot for your attention!