

Business-Prozesse mit der Java Rule Engine API und Drools erstellen



Wer ist der Speaker?

- Frank Schlinkheider
 - 34 Jahre, verheiratet
 - Geschäftsführer der ITSD Consulting GmbH
 - Seit 10 Jahren OO (Smalltalk -> Java -> ??)
 - Schwerpunkte: J2ME und Modellgetriebene SE
- ITSD Consulting GmbH
 - Consulting und Produktentwicklung
 - Objektorientierung, Java (J2ME, J2EE)
 - Banken, Versicherungen, Medien, Metallindustrie (ERA)
 - Sitz in OWL

Agenda

- Einführung in die regelbasierte Softwareentwicklung
- JSR 94
- Drools
 - Drools Language
 - „Domain Specific Languages“
 - Decision Table
- JBoss Rule Workbench
- Zusammenfassung und Diskussion

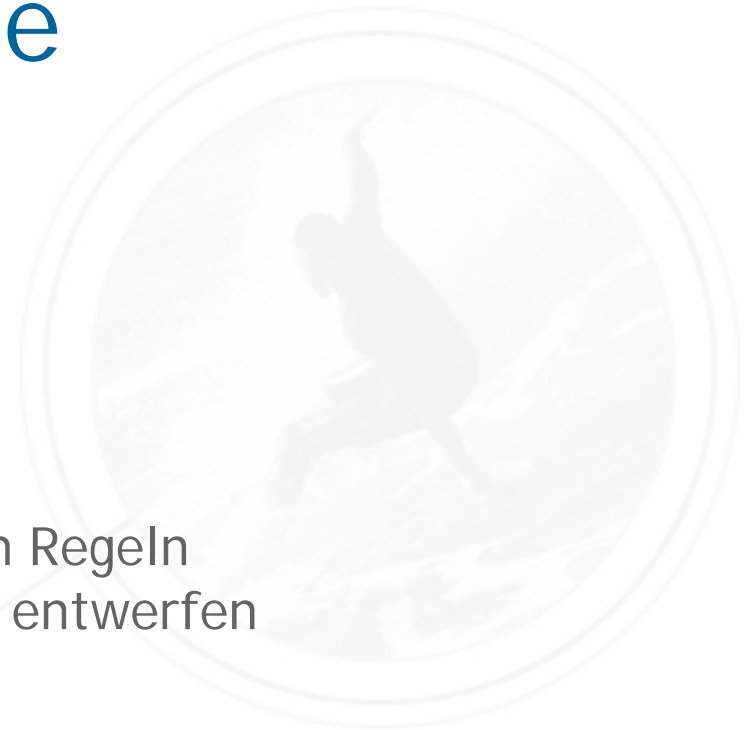


Prinzip

- Auslagerung von Business Prozessen (Regeln)
- Beschreibung der Regeln außerhalb der Programmiersprache
- Trennung von Verarbeitungslogik (Rules) und Daten (Facts)
- Verarbeitung der Regeln durch eine Engine
- Ergebnis: Domainexperten definieren die Anwendungen

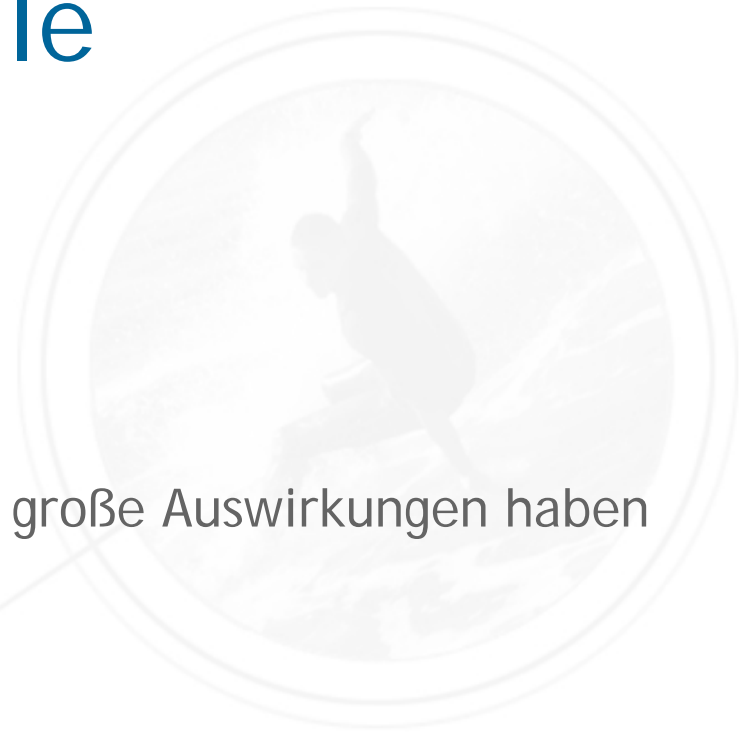
Vorteile

- Bessere Qualität
 - Prozesse werden verbessert
 - Wissen wird „extern“ hinterlegt.
- Höhere Flexibilität
 - Schnelle + einfache Anpassung von Regeln
 - Business Analysten können Regeln entwerfen
 - Einfachere Wartung
- Erhöhte Transparenz
 - Reduzierung von „black-box“ Verhalten in Anwendungen



Nachteile

- evtl. Performance
- Große Abhängigkeiten
 - Eine Änderung in einer Rule, kann große Auswirkungen haben
- „black-box“



Einführung: Was ist eine Rule

- Aufbau - Wenn-Dann-Muster

z.B.: wenn (ich müde bin) dann (gehe ich ins Bett)

- Bedingungen

- arbeiten auf Fakten
- können logisch Verknüpft sein

z.B.: wenn (ich müde bin) und (noch vor 22:00 Uhr)
dann (lege ich mich aufs Sofa)

Einführung: Verkettungen

- Verkettungen von mehreren Regeln
 - wenn er müde ist, geht er ins Bett
 - wenn er ins Bett geht, dann zieht er sich einen Schlafanzug an.
- Vorwärtsverkettung
 $(A \rightarrow B) \wedge (B \rightarrow C) \rightarrow (A \rightarrow C)$
- Rückwärtsverkettung
 - er zieht genau dann einen Schlafanzug an, wenn er ins Bett geht
 - er geht genau dann ins Bett, wenn er müde ist

Einführung: RuleEngine/Fakten

1. Anfangszustand:
„es regnet“ , „warnlicht ist aus“
2. Durchlauf
“es regnet“ , „warnlicht ist aus“ , „strasse ist naß“
3. Durchlauf
„es regnet“ , „warnlicht ist aus“ , „strasse ist naß“ , „strasse ist rutschig“
4. Durchlauf
„es regnet“ , „warnlicht ist an“ , „strasse ist naß“ , „strasse ist rutschig“
5. Endzustand
„es regnet“ , „warnlicht ist an“ , „strasse ist naß“ , „strasse ist rutschig“

Pattern Matching/Agenda

- Welche Regeln sind anwendbar?
- Es existieren viele Bedingungen/Prämissen
- Fakten/Informationen ändern sich
- Änderung einzelner Fakten/Informationen
-> alle Prämissen müssen geprüft werden
- Prämissen sind in mehreren Regeln vorhanden...
- Ineffiziente Auswertungen der Regeln
- Sortierung der auszuwertenden Regeln

Einfacher Pater-Matching-Algo.

```
Für jede Regel {  
  Für jede Prämisse (Teilbedingung): {  
    Information vorhanden, damit diese Prämisse wahr wird?  
  }  
  Wenn alle Prämissen erfüllt sind, dann: {  
    Führe die entsprechenden Aktionen aus!  
  }  
}
```

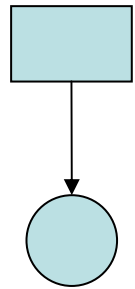
Einführung: Rete Algorithmus

- Charles Forgy, 1982
- Besseres Pattern-Matching
- Nur Bedingungen berücksichtigen, die von Informationsänderungen betroffen sind
- Vorhalten aller Ergebnisse von bereits ausgewerteten Prämissen
- Reduzierung der Prämissen

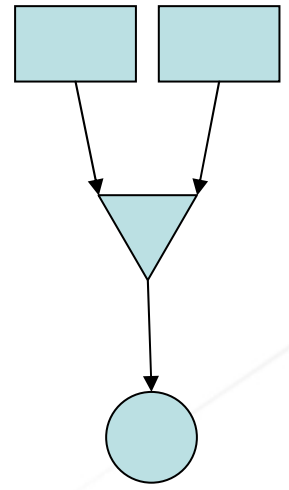
Rete Algorithmus: Knotentypen



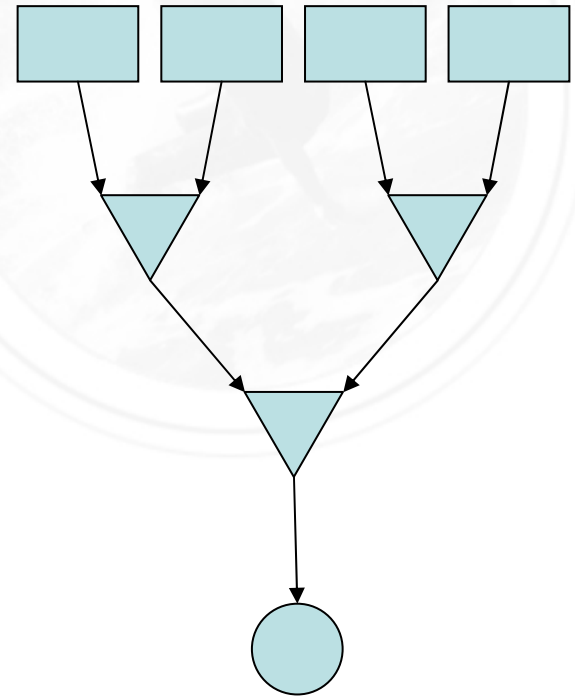
Regel mit 1 Prämisse



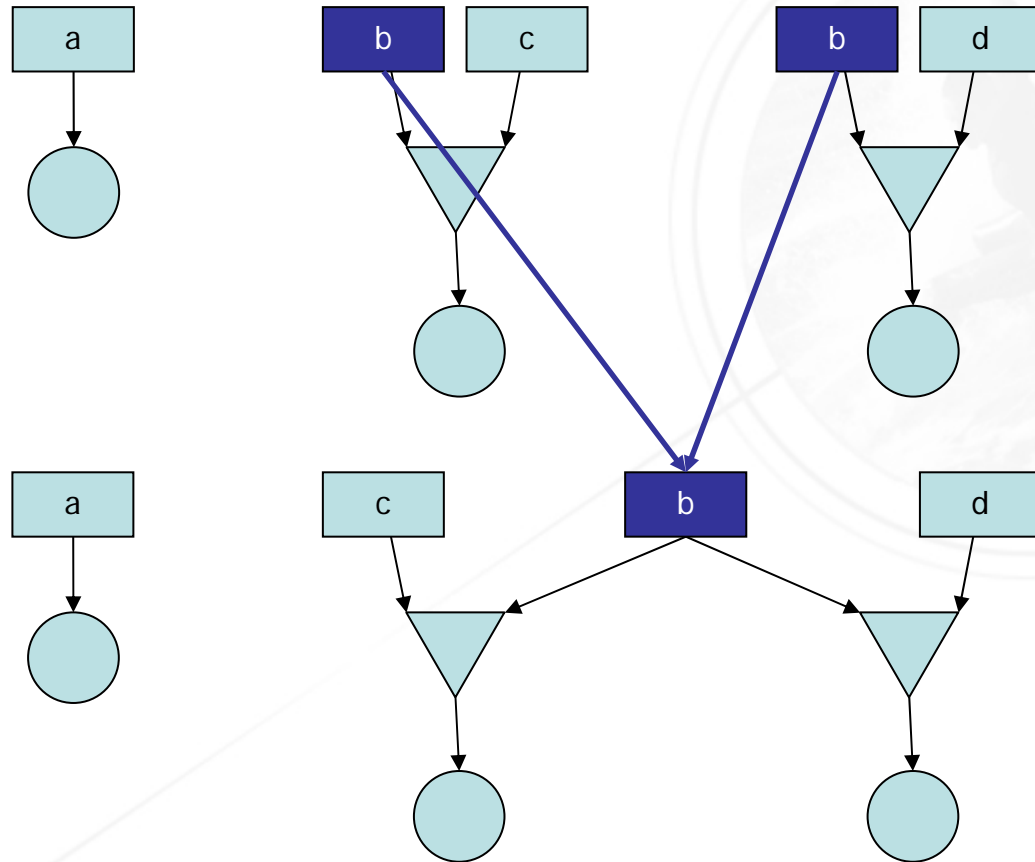
Regel mit 2 Prämissen



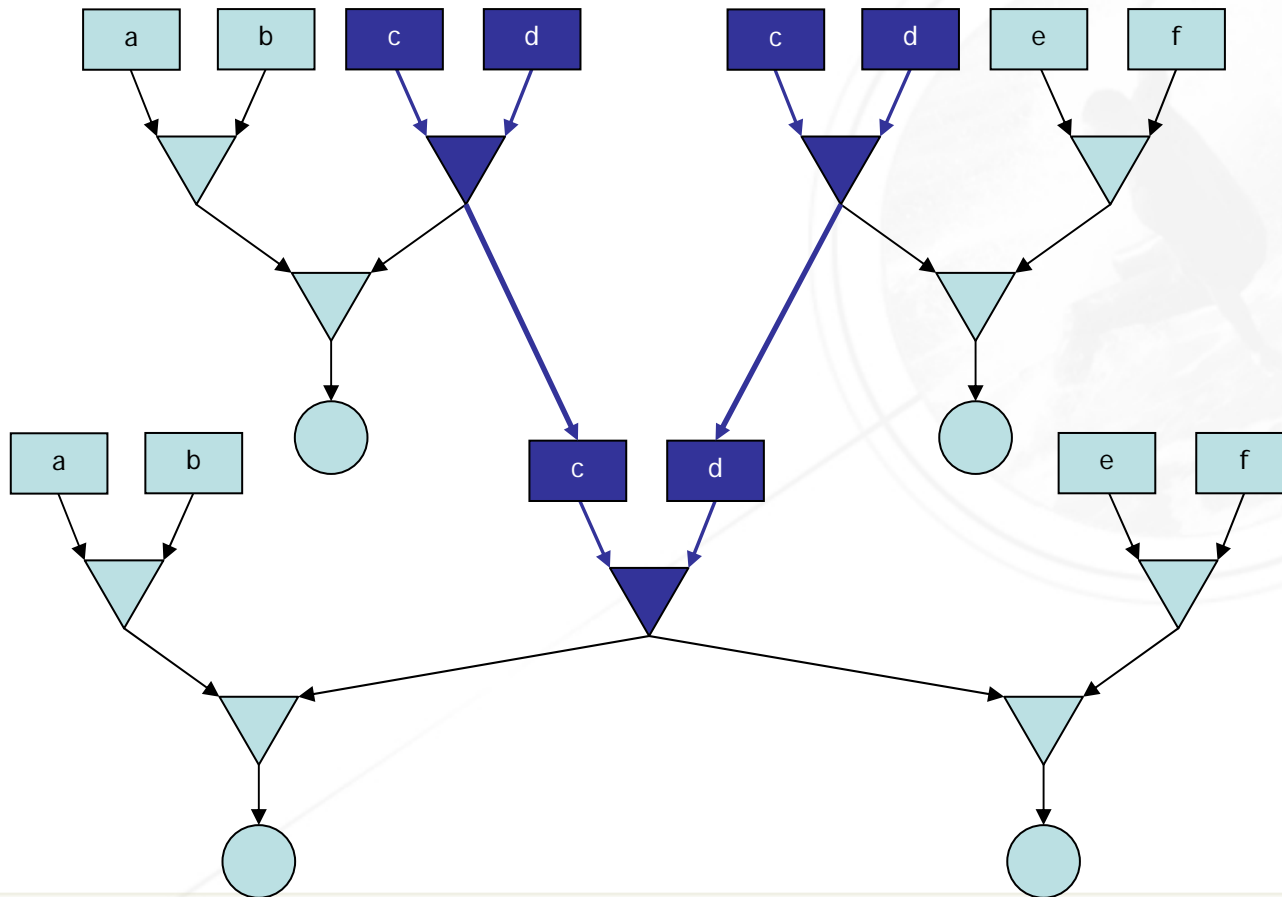
Regel mit 4 Prämissen



Reduktion der Prämissenknoten



Reduktion der Verbindungsknoten



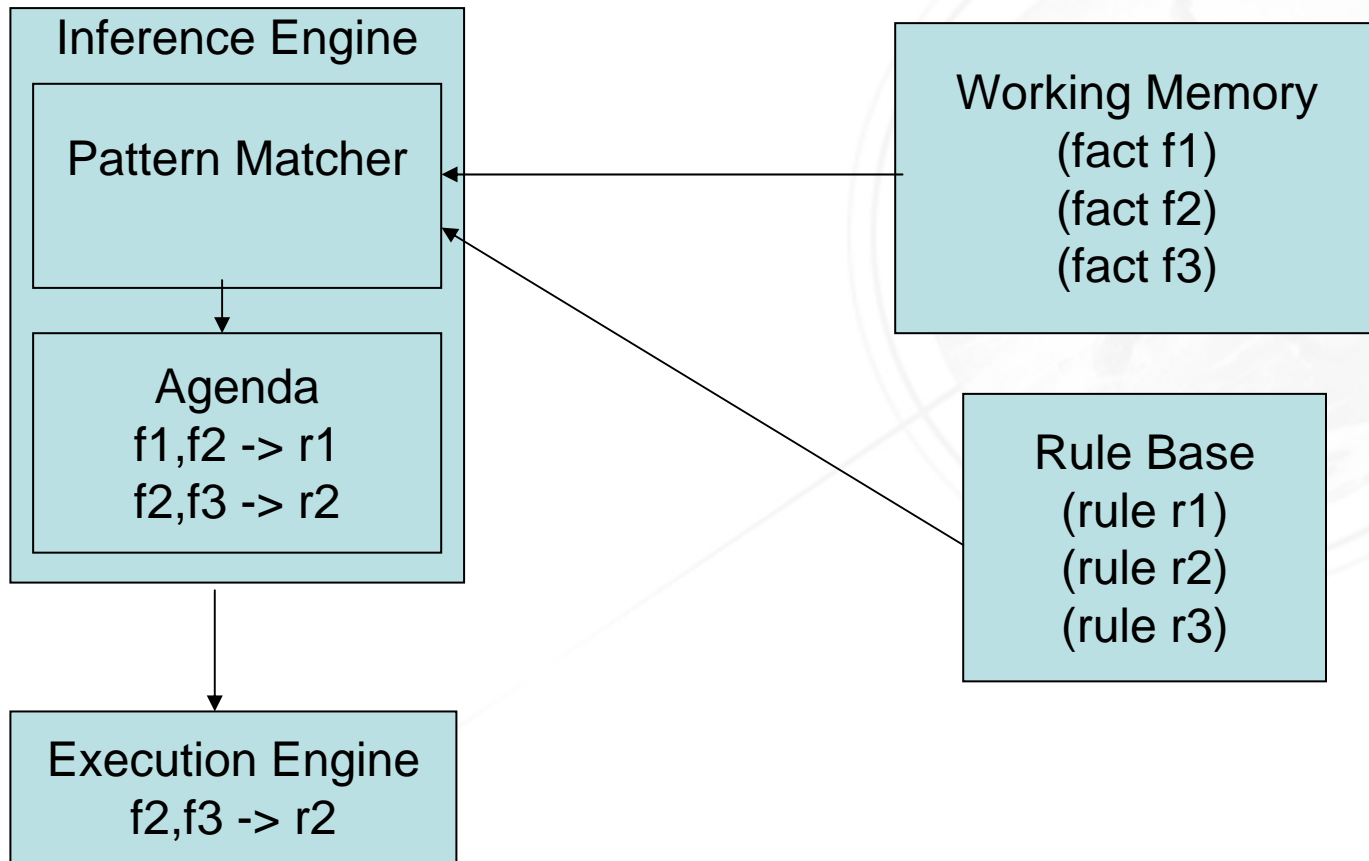
Einführung: Rule Engine 1/4

- Historie: Entwicklungssysteme für Expertensystemen (CLIPS, NASA, 1984)
- Bedingungen
- Fakten
- Aktionen
- Regel:
Wenn Gesamtsumme der Bestellung > 100 € -> 5% Rabatt
 - Bedingung: Wenn Gesamtsumme Bestellung > 100€
 - Aktion: 5%
 - Fakten: z.B. Gesamtsumme: 125€

Einführung: Rule Engine 3/4

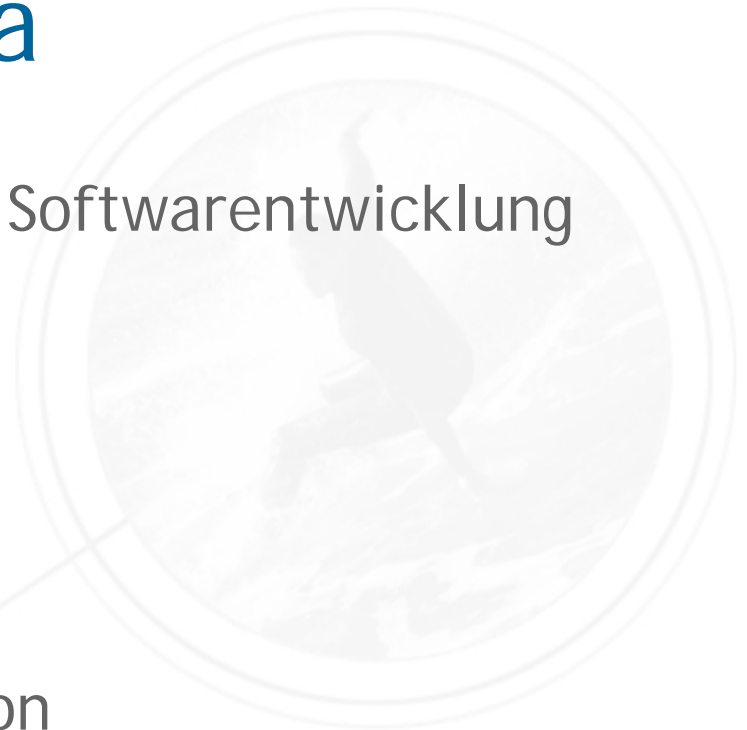
- Working Memory
- Regelinterpretierer
 - Pattern-Matcher
 - Agenda
 - Execution Engine
- Verarbeitung/Interpretation der Rules
- liefert API für die jeweilige Programmiersprache
- Rules werden „extern“ gehalten. z.B. im XML-Dok.
- RuleML
- JSR94 (API-Definition)

Einführung: Rule Engine 4/4



Agenda

- Einführung in die regelbasierte Softwareentwicklung
- JSR 94
- Drools
 - Drools Language
 - „Domain Specific Languages“
 - Decision Table
- JBoss Rule Workbench
- Zusammenfassung und Diskussion

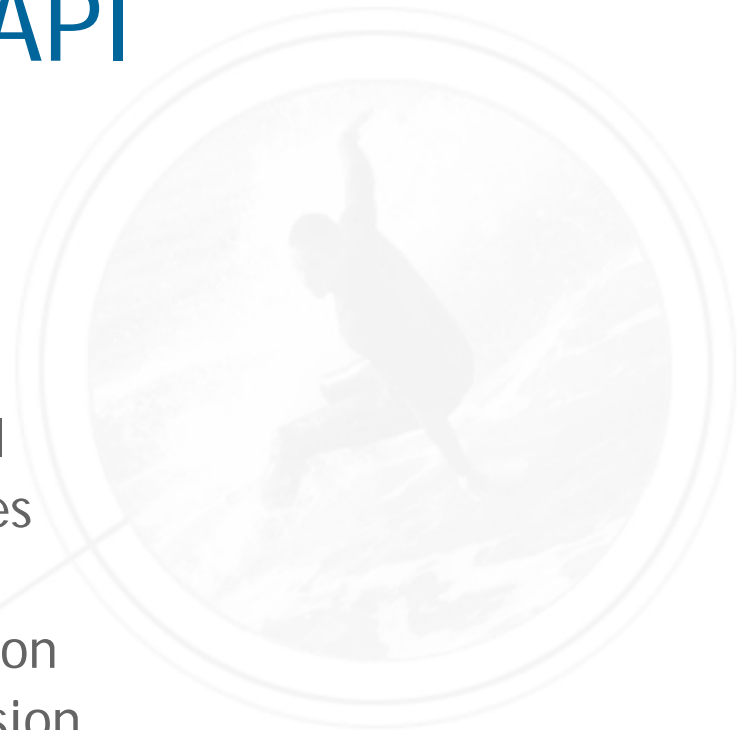


JSR 94

- API-Definition
- Breite Unterstützung (JRules, Jess, Drools, VisualRules?)
- ermöglicht Austausch einer Rule-Engine
- Ablage/Verwaltung von Rules ist nicht spezifiziert
-> Hoher Portierungsaufwand!
- Local/Remotezugriff (JNDI)

JSR 94: API

- Package-Struktur
 - javax.rules
 - javax.rules.admin
- admin.Rule: Beschreibung einer Regel
- admin.RuleExecutionSet: Set von Rules
- RuleSession: Interface
- StatefulRuleSession extends RuleSession
- StatelessRuleSession extends RuleSession
- ObjectFilter: Filter für die Rückgabe
- RuleServiceProviderManager: Verwaltet die RE
- RuleServiceProvider: Abstract; Unterklasse die jeweilige RE-Impl.



JSR 94: Administration 1/3

- Package `javax.rules.admin`
- RuleAdministration: Verwaltung von `RulesExecutionSets`
 - URI: Identifier
- RuleExecutionSetProvider: Einlesen von `RulesExecutionSets`
 - Formate:
 - `org.w3c.dom.Element`
 - `java.io.InputStream`
 - `java.lang.Object`
 - `java.io.Reader`
 - `java.lang.String`

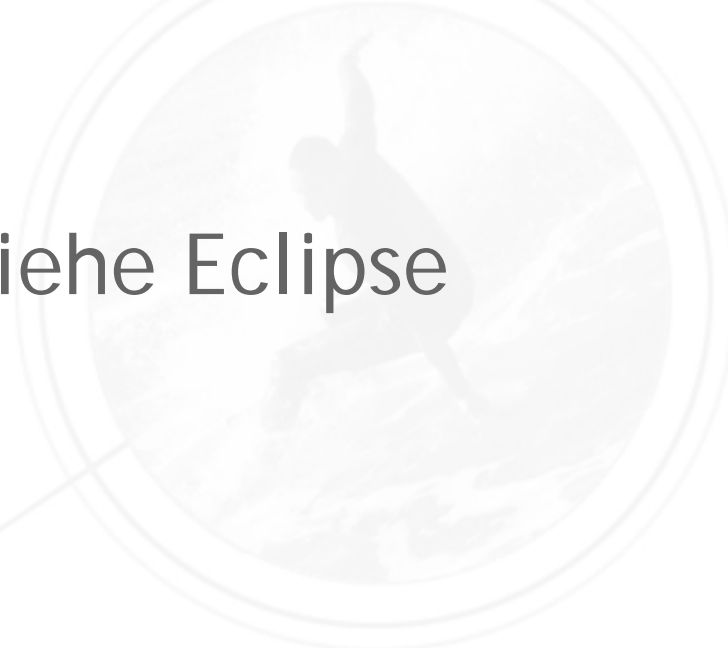
JSR 94: Administration 2/3

- Rule
 - eigentliche Regel
 - Name / Description
 - Rest herstellerabhängig
- RuleExecutionSet
 - Set von Rules
 - Name /Description
 - Set/Get DefaultObjectFilter



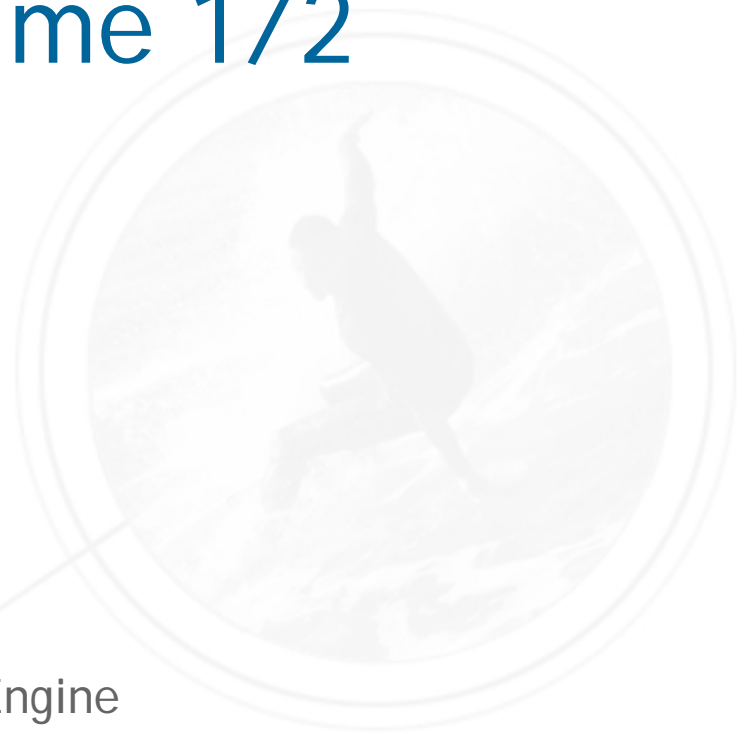
JSR 94: Administration 3/3

Beispiel: siehe Eclipse



JSR 94: Runtime 1/2

- Package javax.rules
- RuleRuntime
 - durch RuleServiceProvider erzeugt
 - liefert die RuleSession
 - RMI-Exception
- RuleSession
 - Connection zwischen Client und RuleEngine
 - eine Session -> eine RuleExecutionSet
 - Systemressourcen
 - Abstract

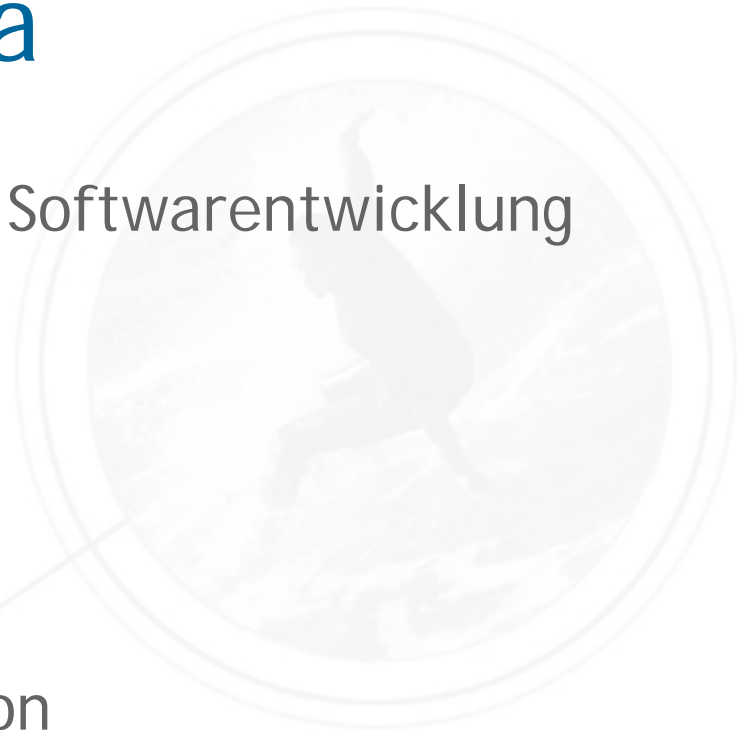


JSR 94: Runtime 2/2

- StatefulRuleSession
 - Interaktion zur RuleEngine möglich
 - Objekte
 - hinzufügen: `Handle addObject(Object)`
 - ändern: `updateObject(Handle, Objekt)`
 - löschen: `void removeObjekt(Handle)`
 - `executeRules();`
- StatelessRuleSession
 - bessere Performance (?)
 - wiederholtes Ausführen -> immer selbes Ergebnis
 - `executeRules(List)`
- Beispiel: siehe Eclipse

Agenda

- Einführung in die regelbasierte Softwareentwicklung
- JSR 94
- Drools
 - Drools Language
 - „Domain Specific Languages“
 - Decision Table
- JBoss Rule Workbench
- Zusammenfassung und Diskussion



Drools

- Open Source, Apache Software Foundation-Lizenz
- seit Version 3.0 JBoss Rules (April 2006)
- herstellerabhängige API
- JSR 94 konform
- Eclipse Plugin
- neue Rule Language (seit 3.0)
- Rule File: *.drl
- Domain Specific Language *.dsl



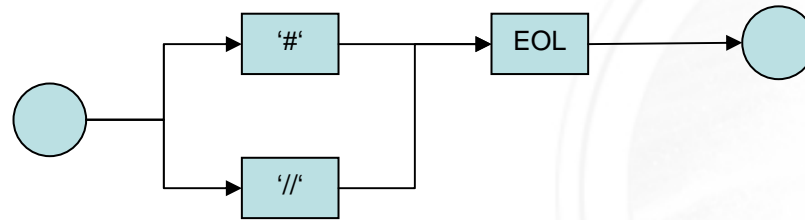
Drools: Rule Language

- Beschreibung der Regeln
- kein XML-File
- Unterstützung von umgangssprachlichen Formulierungen
-> Domain Specific Language
- Grundstruktur einer Regel:

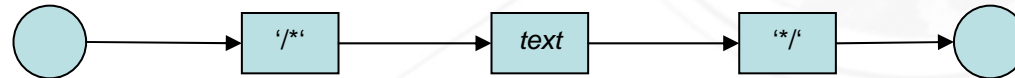
```
rule "name"  
    ATTRIBUTES  
when  
    LHS  
then  
    RHS  
end
```

Drools Language: Comments

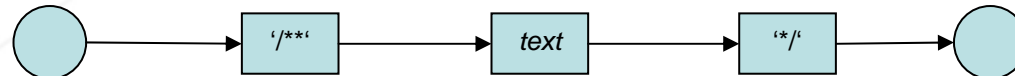
Single line comment



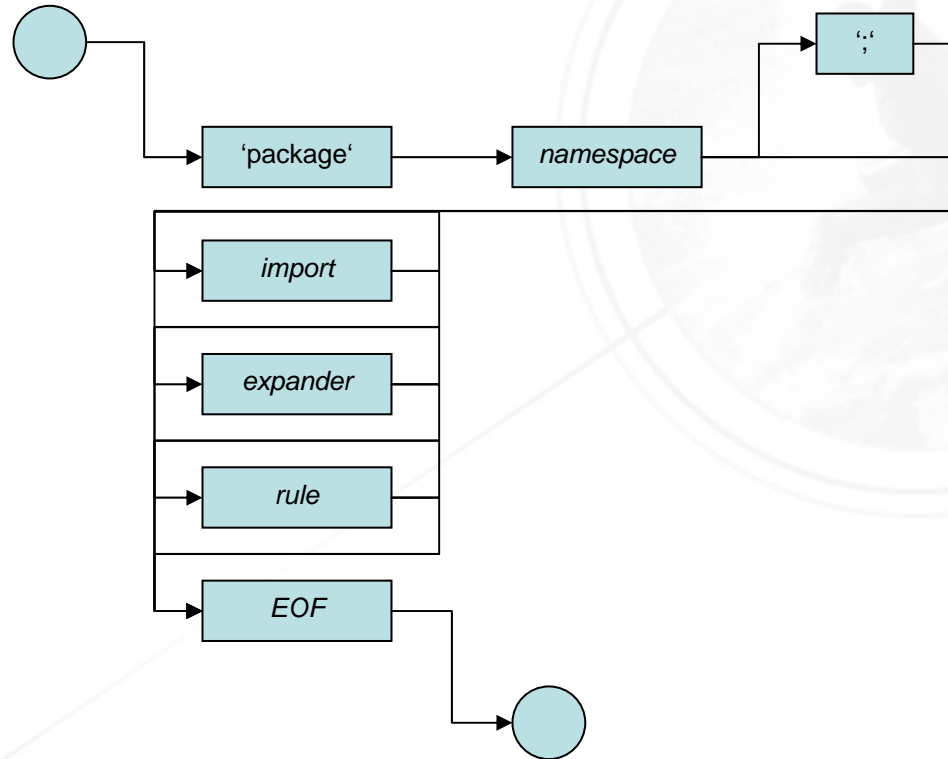
Multi line comment



Documentation

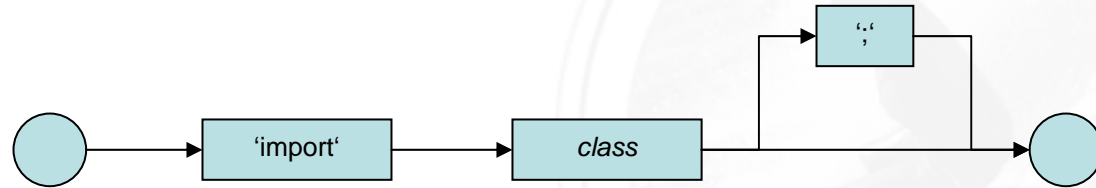


Drools Language: package

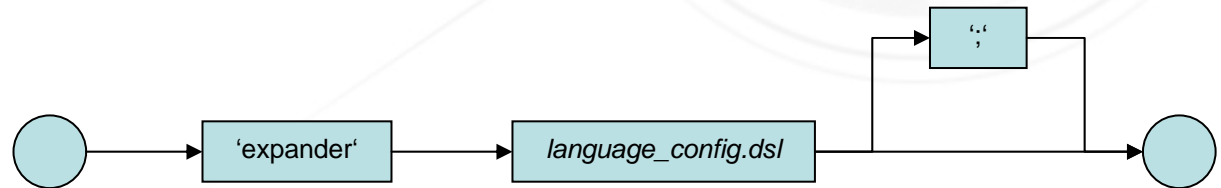


Drools Language: import, expander

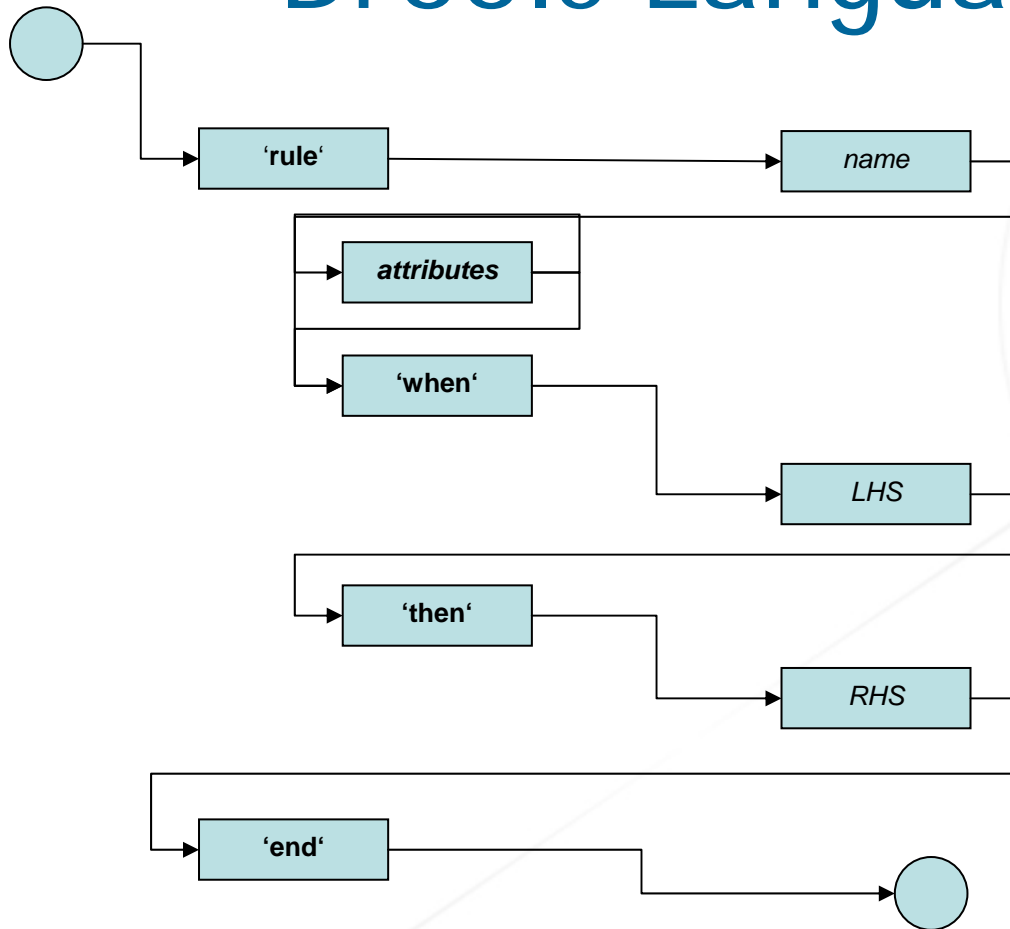
import



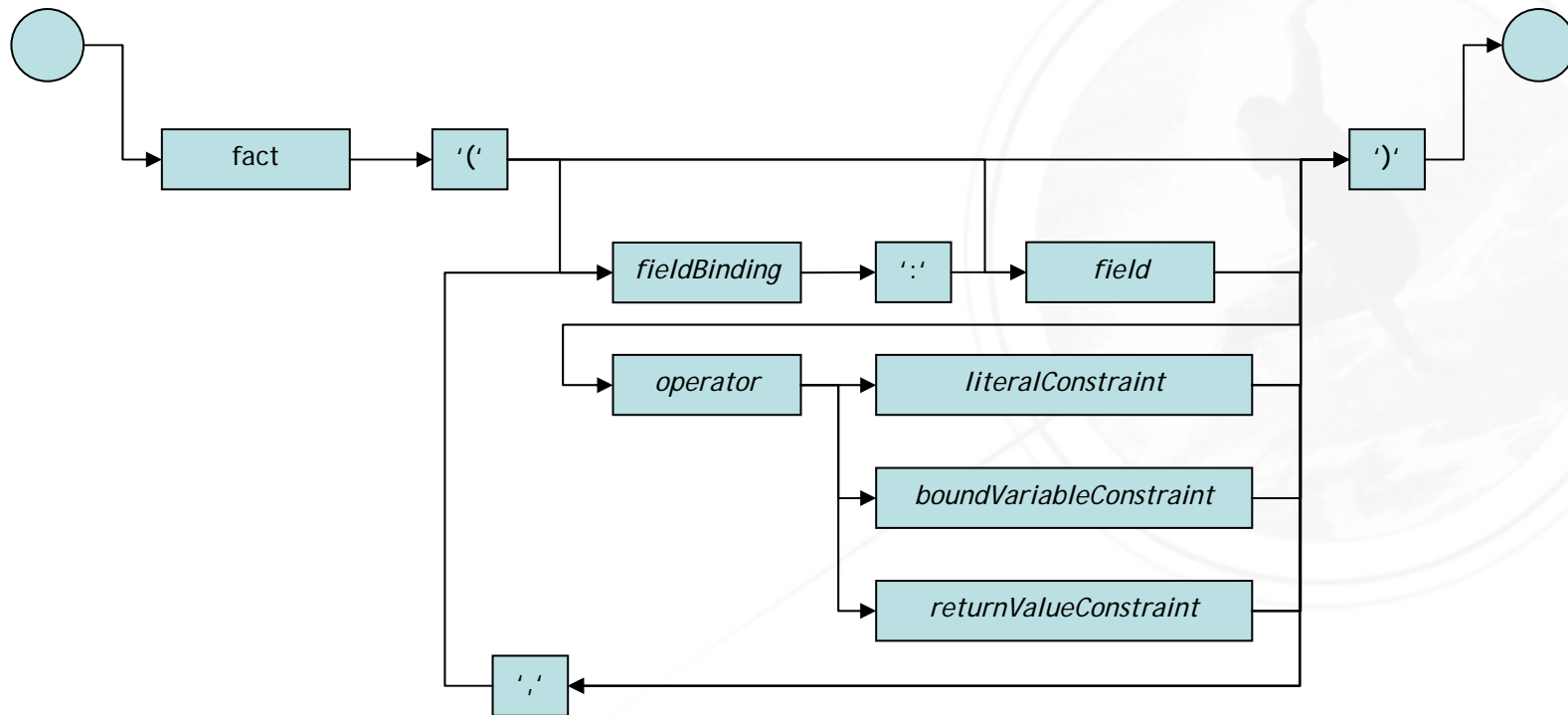
expander



Drools Language: rule



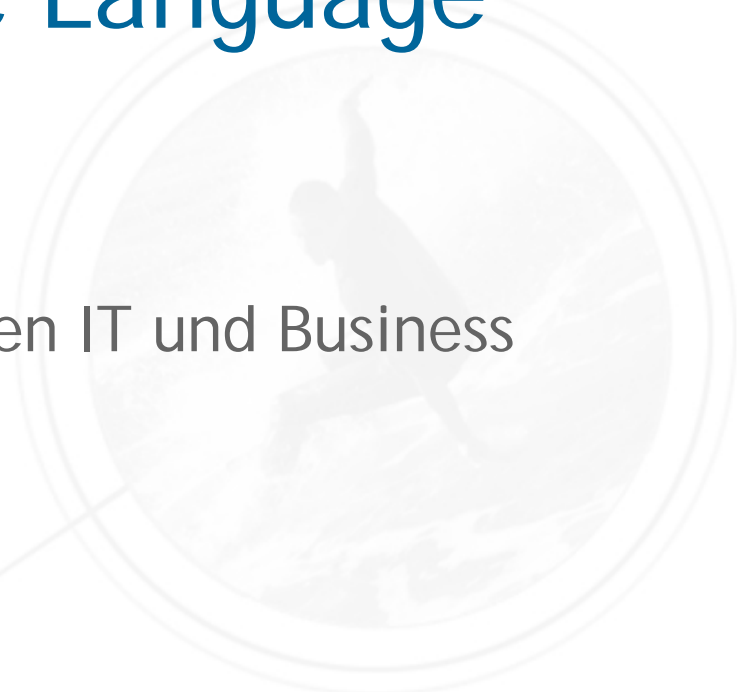
Drools Language: Column



- *Bsp.: Auftrag(status == „gebucht“, summe > 100)*

„Domain Specific Language“

- natürliche Sprache
- Mapping
- bessere Kommunikation zwischen IT und Business Analyst
- separate Files *.dsl



Example: DSL/DRL

```
package SistersRules
import org.drools.jsr94.rules.Person;
expander sisters_expander.dsl

rule "FindSisters"
when
    #conditions
    There is a Person
    There is a second Person and they are sisters
then
    Leave a comment in the WorkingMemory
end
```



Example: DSL

[when]

There is a Person=

```
$person1 : Person ( )
```

[when]

There is a second Person and they are sisters=

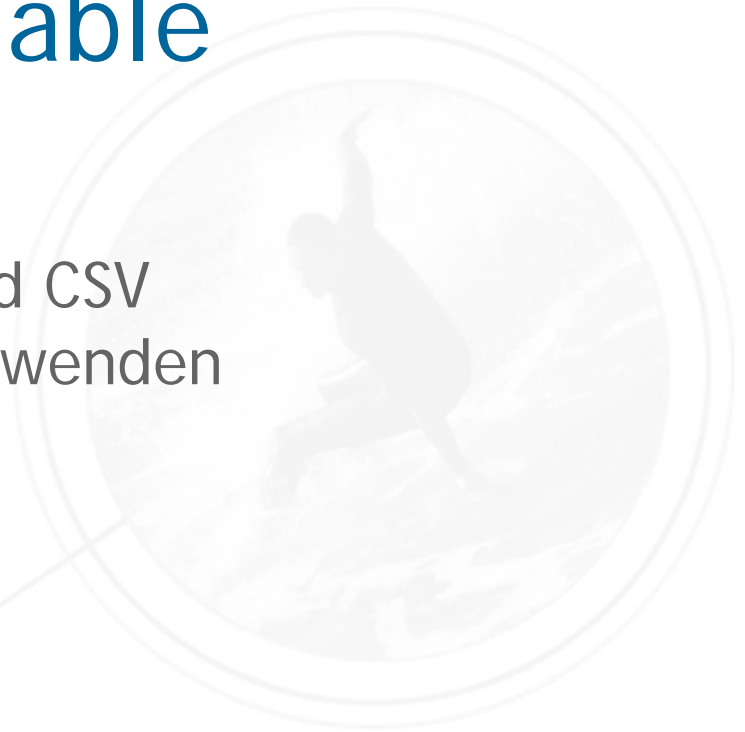
```
$person2 : Person ( ) eval( $person2.hasSister($person1) )
```

[then]Tell the world they are sisters=

```
System.out.println($person1.getName() + " and "  
+ $person2.getName() +" are sisters");
```

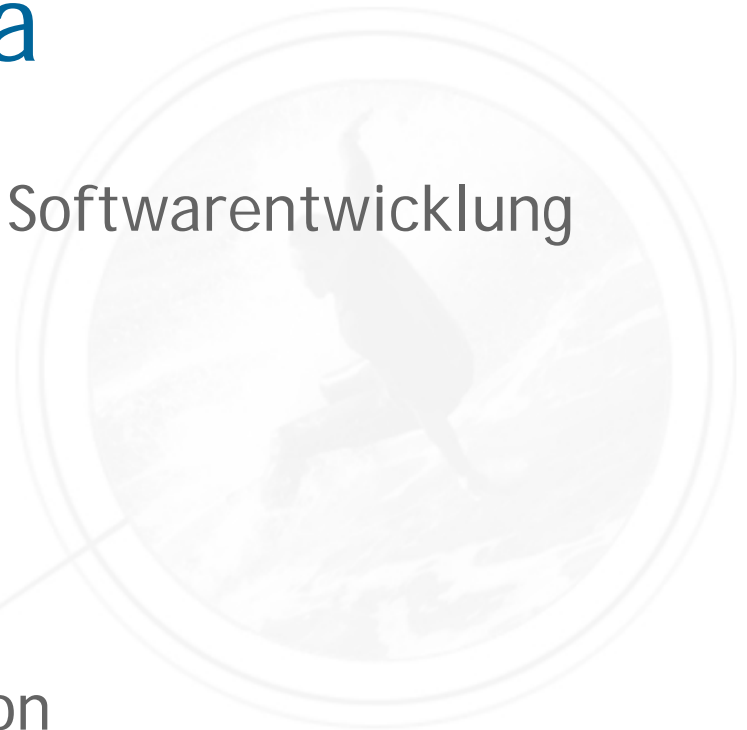
Decision Table

- Alternativformate
- Microsoft Excel, OpenOffice und CSV
- Business Analyst kann Excel verwenden
- transparent für den Entwickler
- Example: siehe Eclipse



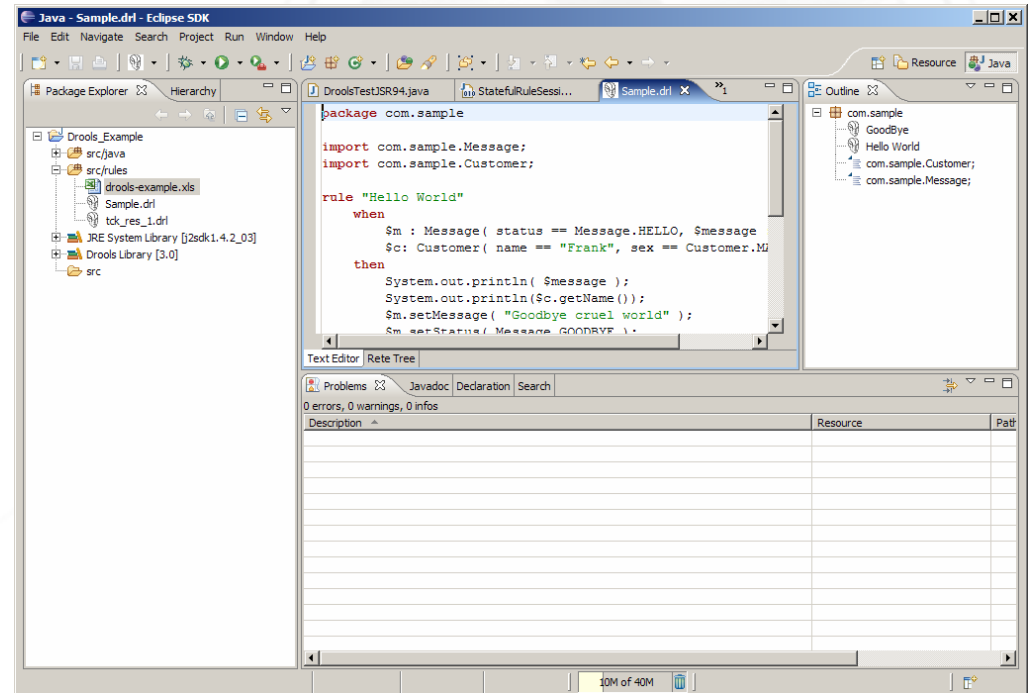
Agenda

- Einführung in die regelbasierte Softwareentwicklung
- JSR 94
- Drools
 - Drools Language
 - „Domain Specific Languages“
 - Decision Table
- JBoss Rule Workbench
- Zusammenfassung und Diskussion



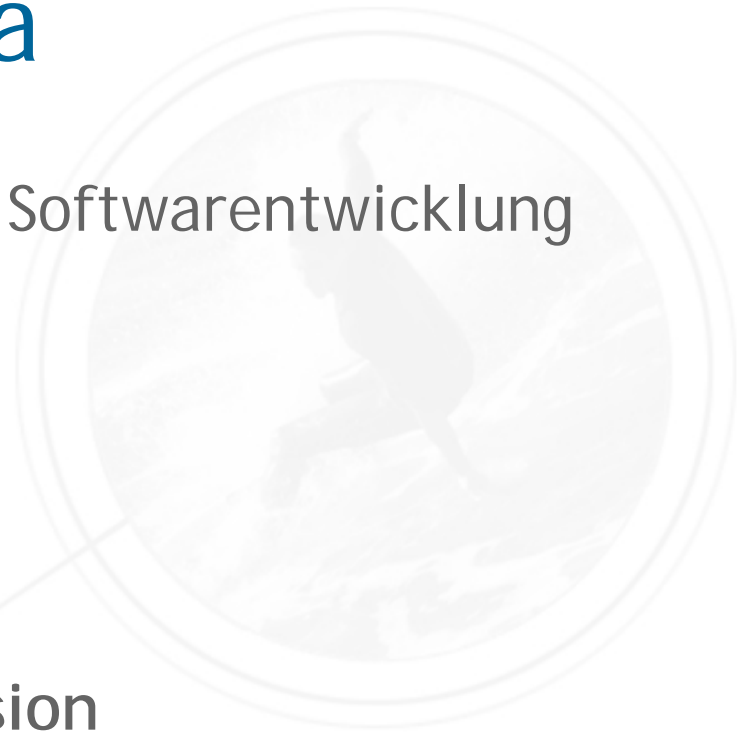
JBoss Rule Workbench

- Eclipse-Plugin (zum selber bauen...)
- Syntax Highlighting
- Code-Assistenten
- Wizards
 - Projekt
 - Rule
 - Domain Specific Language
 - Decision Table
- Rule Validierung und Error-Reporting
- Example: siehe Eclipse



Agenda

- Einführung in die regelbasierte Softwareentwicklung
- JSR 94
- Drools
 - Drools Language
 - „Domain Specific Languages“
 - Decision Table
- JBoss Rule Workbench
- Zusammenfassung und Diskussion



Zusammenfassung: RuleEngines

- Auslagerung von Business Prozessen
- Verarbeitung der Regeln durch eine Engine
- Beschreibung der Regeln außerhalb der Programmiersprache -> Regeln können von Business Analysten entworfen werden
- Rete-Algorithmus verbessert Performance
- Regelbeschreibung ist schwierig

Zusammenfassung: JSR94

- Einheitlich API
 - Admin
 - Runtime
- Beschreibung der Regeln sind herstellerabhängig
-> großer Portierungsaufwand
- Breite Unterstützung (Drools/JBoss, Jess, ILOG, etc.)
- Weitere Kapselung ist wichtig

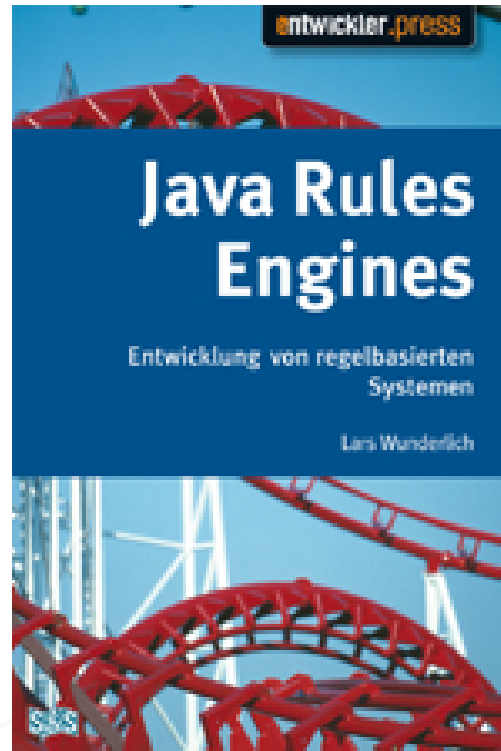
Zusammenfassung: JBoss Rules

- OpenSource
- JBoss Rule Workbench
- JSR 94
- Neue Rule Definition Language
- Excel, OpenOffice und CVS möglich
- „noch“ schlechte Dokumentation

Wichtige URLs

- JSR 94
<http://www.jcp.org/en/jsr/detail?id=94>
- JBoss Rules
<http://www.jboss.com/products/rules>
- Drools
<http://www.drools.org>
- ITSD Consulting GmbH
<http://www.itsd-consulting.de>
- Bachelorarbeit von Hendrik Beck
siehe Google

Buchempfehlung



Fragen ?

Frank Schlinkheider • Enterprise Web & Mobile Solutions
fs@itsd-consulting.de
www.itsd-consulting.de