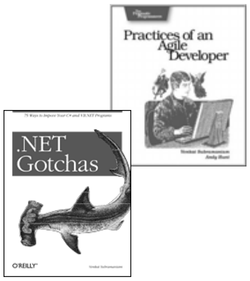
Working with Rule Engines

Venkat Subramaniam

```
spkr.name = 'Dr. Venkat Subramaniam'
spkr.founder = 'Agile Developer, Inc.'
spkr.affiliated = 'University of Houston'
spkr.associated = 'Rice Univ. Continuing Studies'

spkr.cred = %w{Programmer Speaker Trainer Author}

spkr.nfjs = 'NFJS Speaker since 2002'

spkr.website = 'http://www.agiledeveloper.com'
spkr.email = 'venkats@agiledeveloper.com'
```



Abstract

Rule based programming allows us to develop applications using declarative rules. These can simplify development in applications where such rules based knowledge is used for decision making.

In this presentation we will take a look at the tools and techniques for developing rule based applications. We will take a look at open source tools, discuss their strengths, capabilities, and limitations.

Working with Rule Engines

- Rule Engines
- Programming Model
- JSR 94
- Rules
- Tools and Drools
- Conclusion

Rule

- Rule: "Defines or constraints some aspect“
- Rules inference and decision making is critical
  - Insurance, intelligence, taxation, finance, expert systems
- Mail filter (we’ve all configured rules)
- Store discount qualification, amount
- Tax determination by state/region
- State regulations in product sales

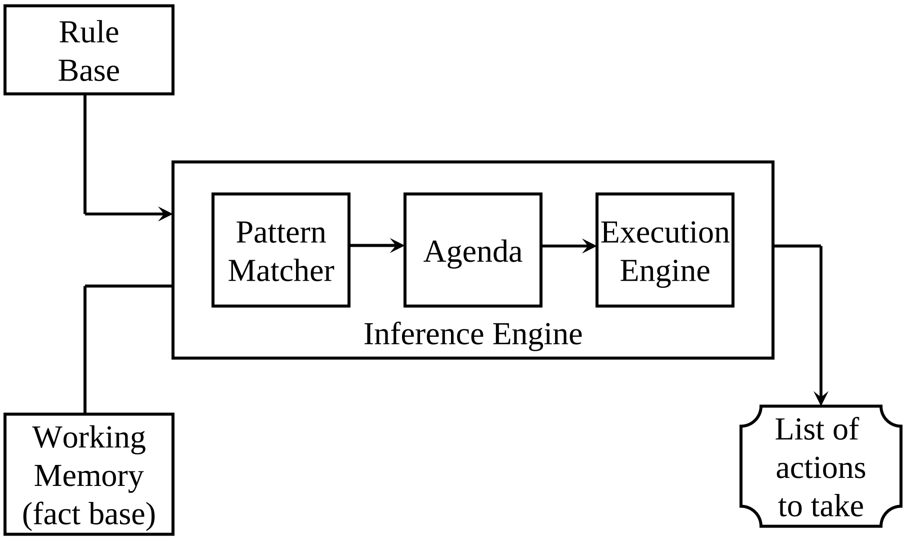
Concerns

- But there is one big concern
  - That is separation of concern
- Having business rules mixed with rest of your code is not wise
  - Each time the data and rules around it change, you don't want to change code
  - It becomes cost prohibitive to keep up with such change
- Need to separate knowledge from its implementation
  - Change in knowledge does not require change to source code

Rule Engines

- Rule Engine allows you to separate the business logic
  - Evaluates, infers, and executes rules
- Rules expresses as high level if-else statements
  - Has a condition (if) part and an action (then) part
  - The order in which rules are specified is not important
  - They act collectively

Rule Engines



Working with Rule Engines

- Rule Engines
- Programming Model
- JSR 94
- Rules
- Tools and Drools
- Conclusion

Programming Model

- Procedural programming
  - You provide sequence of instructions to execute
  - You show each step to follow, that is, you show how to do stuff
  - Gets too cumbersome for rule based systems

- Declarative Programming
  - Rule-based programming is declarative
  - You say what you want, not how to do it
  - A bit hard since we're not used to it
  - Inductive and very powerful, however
  - If you have used XSLT, you've done it
  - You say what you want, then delegate to the rule engine to evaluate and infer

---

Working with Rule Engines

- Rule Engines
- Programming Model
- JSR 94
- Rules
- Tools and Drools
- Conclusion

JSR 94

- JCP process
- Java runtime API for rules engines
- Standardization of rules engine interface
- You can plug-in different rule engines into your runtime
- Facilitates creating and deploying rules
- Does not define language to define rules and actions
- Something like RuleML may help in this effort

API

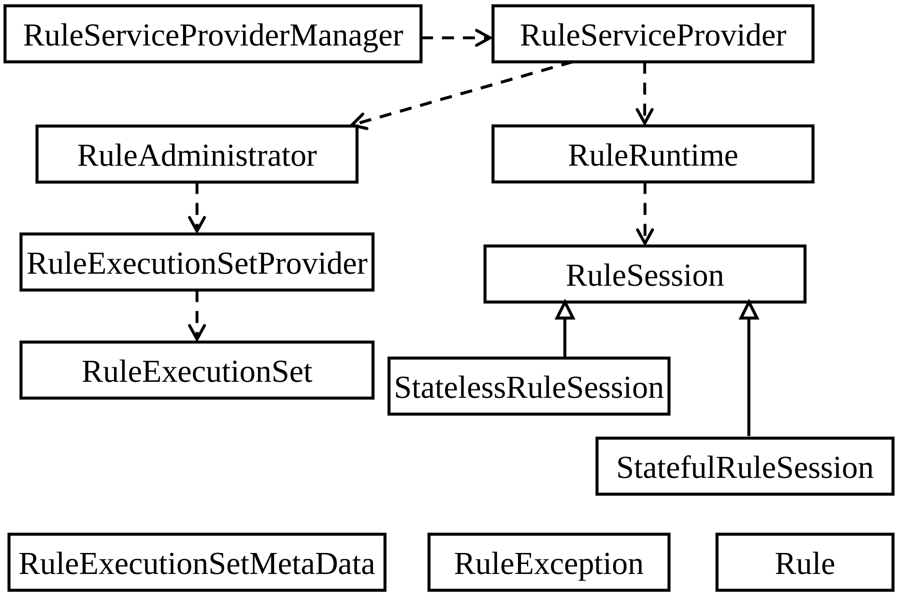
- Clearer separation of API to administer and API to use rules

- Rules Administration API: javax.rules.admin
  - Classes to load rules and associate actions
  - Options to load rules from different resources
  - Register and unregister rule execution sets
  - Authorization support

- Runtime Client API: javax.rules
  - Classes for clients to execute rules within a session

- Rule Execution Set Metadata provides details about the rule execution set

Classes and Interfaces



RuleServiceProviderManager

RuleServiceProvider

RuleAdministrator

RuleRuntime

RuleExecutionSetProvider

RuleSession

RuleExecutionSet

StatelessRuleSession

StatefulRuleSession

RuleExecutionSetMetaData

RuleException

Rule

Rule Session

- Stateless Rule Sessions
  - The execution and evaluation is on a per request basis
  - You can send either one object or a list of objects for evaluation
- Stateful Rule Sessions
  - Remembers the context
  - You can add objects
  - You can update objects

Working with Rule Engines

- Rule Engines
- Programming Model
- JSR 94
- Rules
- Tools and Drools
- Conclusion

Facts, Patterns, and Rules

- Facts
  - A piece of information
  - are tuples with no variables
- Patterns are tuples with variables
- Rules have left-hand side lists of patterns

Patterns (predicates/premises) => actions/conclusions

It's a fact

- Rule evaluation involves inferring fact
- Facts are goals that are true and have no sub-goals

- Which facts need to be true for this rule to be fired?
  - Backward chaining
- Give facts which rules are fired?
  - Forward Chaining

- Backward Chaining
  - You break a goal into sub-goal and try to prove those recursively until you reach facts
  - Should I take an action? I'll check the facts
  - Prolog takes this approach

- Forward chaining
  - A data driven approach where you derive goals from facts
  - If facts are true, then take action
  - CLIPS takes this approach
  - RETE algorithm

RETE Algorithm

- So many rules, so many facts, so much complexity
  - naïve algorithm => poor performance

- RETE (ree-tee) Charles L. Forgy created it
- Algorithm to improve speed of forward chaining
  - Works effectively by limiting the effort to recompute nodes when rules are fired
  - Works on assumption that relatives few objects change when a rule is fired
  - State machine based approach on Rules expresses as data
  - However, high memory usage
    - Space compromised for speed

- RETE creates a decision tree that combines patterns in all the rules
- The tree of nodes or network
  - RETE is Latin for network

Working with Rule Engines

- Rule Engines
- Programming Model
- JSR 94
- Rules
- Tools and Drools
- Conclusion

Tools

- OPSJ
- iLog,
- JRules
- Blaze Advisor
- Jess
- Mandarax
- Drolls
- ...

JESS

- Java Expert System Shell
  - Based on CLIPS Expert System Shell C Language Integrated Production System
  - LISP like syntax
  - Rules - if/else conditions
  - Knowledgebase - data on which rules applied
  - Takes a state, applies rules and arrives at a state
- Typically used as an embedded system in your application
- Also provides direct access, command line, GUI, and servlet
- Not open source (even though used as reference implementation)

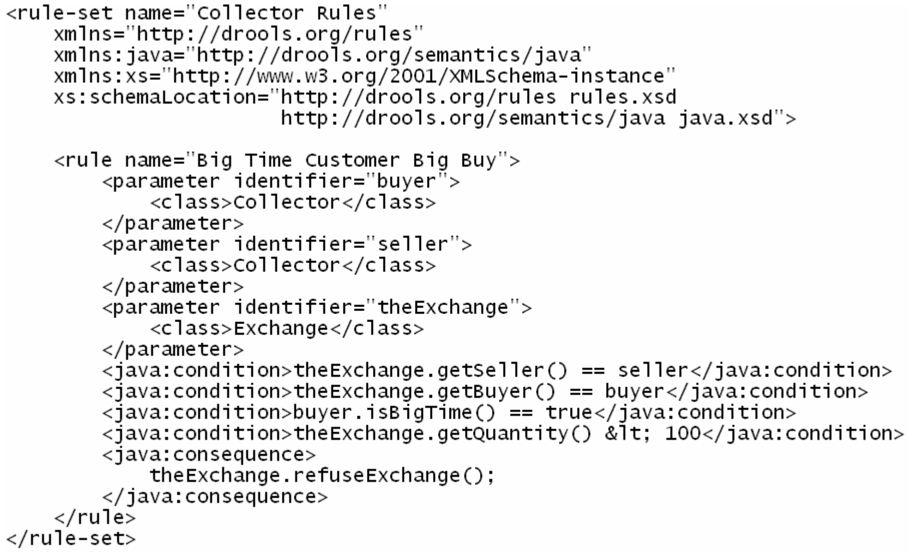
Drools

- JSR 94 compliant
- XML syntax
- Open Source
- Easy to express rules
- Nice integration as well
- Rules expressed in XML file
  - With embedded Java code if you desire
  - May also host other languages
  - Groovy, Python, DSL
- Conflict Resolution
- You’re Drooling for it?

DRL File

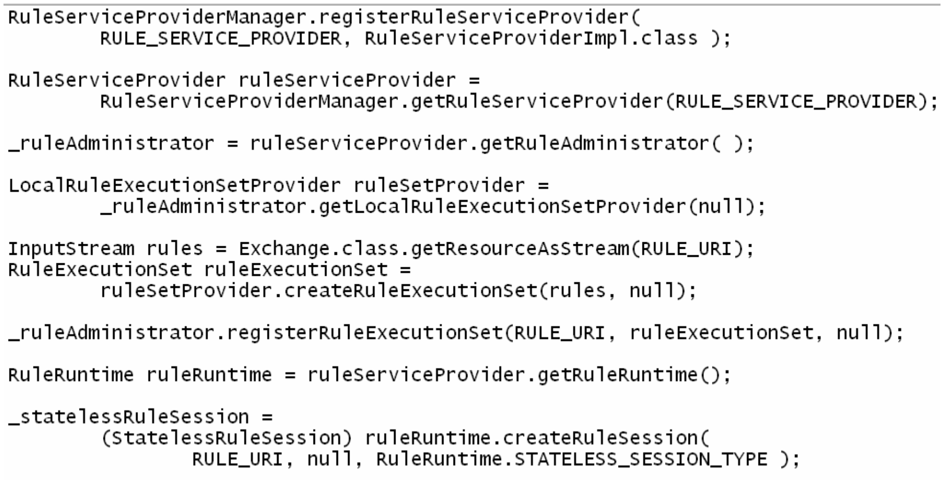
- Use to specify the rules
- You express your rule set here
- Basic Semantic Model
- Language-specific Semantic Model
  - Java, Groovy, Python, Domain Specific Lang.

Example drl File



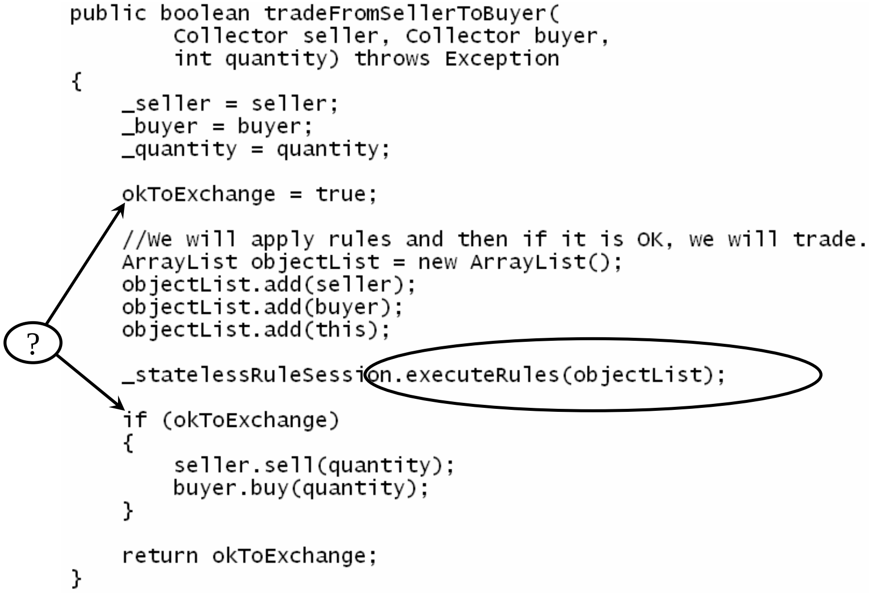
```
<rule-set name="Collector Rules"
    xmlns="http://drools.org/rules"
    xmlns:java="http://drools.org/semantics/java"
    xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
    xs:schemaLocation="http://drools.org/rules rules.xsd
                       http://drools.org/semantics/java java.xsd">

    <rule name="Big Time Customer Big Buy">
        <parameter identifier="buyer">
            <class>Collector</class>
        </parameter>
        <parameter identifier="seller">
            <class>Collector</class>
        </parameter>
        <parameter identifier="theExchange">
            <class>Exchange</class>
        </parameter>
        <java:condition>theExchange.getSeller() == seller</java:condition>
        <java:condition>theExchange.getBuyer() == buyer</java:condition>
        <java:condition>buyer.isBigTime() == true</java:condition>
        <java:condition>theExchange.getQuantity() &lt; 100</java:condition>
        <java:consequence>
            theExchange.refuseExchange();
        </java:consequence>
    </rule>
</rule-set>
```

Setting up – the JSR 94 way



```
RuleServiceProviderManager.registerRuleServiceProvider(
        RULE_SERVICE_PROVIDER, RuleServiceProviderImpl.class );

RuleServiceProvider ruleServiceProvider =
        RuleServiceProviderManager.getRuleServiceProvider(RULE_SERVICE_PROVIDER);

_ruleAdministrator = ruleServiceProvider.getRuleAdministrator( );

LocalRuleExecutionSetProvider ruleSetProvider =
        _ruleAdministrator.getLocalRuleExecutionSetProvider(null);

InputStream rules = Exchange.class.getResourceAsStream(RULE_URI);
RuleExecutionSet ruleExecutionSet =
        ruleSetProvider.createRuleExecutionSet(rules, null);

_ruleAdministrator.registerRuleExecutionSet(RULE_URI, ruleExecutionSet, null);

RuleRuntime ruleRuntime = ruleServiceProvider.getRuleRuntime();

_statelessRuleSession =
        (StatelessRuleSession) ruleRuntime.createRuleSession(
                RULE_URI, null, RuleRuntime.STATELESS_SESSION_TYPE );
```

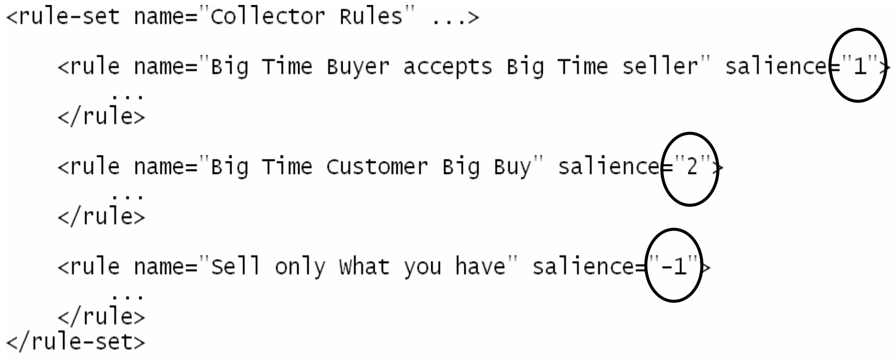
Firing the Rules



```
public boolean tradeFromSellerToBuyer(
        Collector seller, Collector buyer,
        int quantity) throws Exception
{
    _seller = seller;
    _buyer = buyer;
    _quantity = quantity;

    okToExchange = true;

    //We will apply rules and then if it is OK, we will trade.
    ArrayList objectList = new ArrayList();
    objectList.add(seller);
    objectList.add(buyer);
    objectList.add(this);

    _statelessRuleSession.executeRules(objectList);

    if (okToExchange)
    {
        seller.sell(quantity);
        buyer.buy(quantity);
    }

    return okToExchange;
}
```

Conflict Resolution

- What's the order in which rules applied?
  - Salience – priority
  - Recency – based on usage
  - Complexity – priority for more complicated
  - LoadOrder – order in which rules loaded



```
<rule-set name="Collector Rules" ...>

    <rule name="Big Time Buyer accepts Big Time seller" salience="1">
        ...
    </rule>

    <rule name="Big Time Customer Big Buy" salience="2">
        ...
    </rule>

    <rule name="Sell only what you have" salience="-1">
        ...
    </rule>
</rule-set>
```

DSL

- XML syntax included Java code
- Good in a way, pretty bad in another
- How about expressing in a easier way
- You can create your own Domain Specific syntax and use it
  - Define an XSD with constraints
  - Create Factories for conditions and actions
  - Configure it

Neat, but can you make it simpler?

- Making it easier for business people
- How about Excel spread sheet?
- Specify conditions, consequence in Excel
  - or any spreadsheet that can create csv file

Quiz Time

Working with Rule Engines

- Rule Engines
- Programming Model
- JSR 94
- Rules
- Tools and Drools
- Conclusion

Conclusion

- Pros
  - Easy to understand
  - Good separation
  - Easy to keep up with dynamic change
  - Open Source tools

- Cons
  - Learning curve
  - Not for every problem you have
    - Need to carefully pick problems for the tool
  - Dependence on yet another tool
    - the rule engine

References...

- http://www.drools.org
- http://www.drools.org/Decision+Tables
- http://www.jcp.org/aboutJava/communityprocess/review/jsr094
- http://java-source.net/open-source/rule-engines
- https://groovyrules.dev.java.net/
- Paul Browne, “Give Your Business Logic a Framework with Drools,” http://www.onjava.com/pub/a/onjava/2005/08/03/drools.html
- Jess in Action : Java Rule-Based Systems by Ernest Friedman-Hill
- "Rete: A Fast Algorithm for the Many Pattern/ Many Object Pattern Match Problem,“ Charles L. Forgy, Artificial Intelligence 19 (1982), 17-37.

Download examples/slides from

http://www.agiledeveloper.com/download.aspx