BUSINESS RULES MANAGEMENT AND BPM

WHO'S MANAGING YOUR RULES?

Paul Vincent
Rules Specialist and Product Management
Fair Isaac
Agenda

- Business Rules Approach
  – a quick overview

- Rule Engines and Rule Management
  - past and present

- Business Rule Management
  - what it means for IT

- Business Rules vs Business Processes
  - the differences in management

- Case Studies
  - who uses this stuff and why

- Q&A & Further Reading
Quick Overview: What is “the Business Rules Approach”?
Business Rules Approach

- Business rules should be defined, stored, reported, etc **separately** from other entities
  - In business documentation
  - In business process definitions
  - In use cases and system requirements
  - In code
- Cf “data management”
  - **Data** is defined and managed separately as a **common practice**
- Versus “objects”
  - Objects should **encapsulate** data, behavior
  - … but does not prevent us managing such data + rules separately
Business Rules Approach versus Others

- Why not keep the rules in (their original) business documents, policy manuals etc?
  - Difficult to compare, correlate, redefine, verify, exploit, enforce etc

- Why not embed the rules in our data representations?
  - Data schema not a “natural” representation for many rule types

- Why not embed rules in “code” (VB, Javascript, EJBs, Java, C++ etc)?
  - Very flexible for any time of rule definition
  - Coder controls rule definition and execution
    - Rule interrelationships get tricky
  - “Legacy code”: what relationship is there between business and implemented systems?
Main Drivers for using Business Rules Approach

1. Centralize / standard rule execution / management strategy in an organization
   → always know what rules are where

2. Apply rules in a standard way across channels / subsidiaries
   → always use the same rules from the same source

3. Allow businesses to control what rules are executed, and update them as required
   → return business decision control to the business and allow for timely changes to IT systems

4. Allow more complex processes to be automated
   → allow for planning, scheduling, best-choice type decisions to be made
“Rule Engines“ and “Rules Management”

past and present
Rule based Programming Tools

- **Component-based Rule engines**: eg Blaze Advisor, JESS, JRules, Aion
- **OO Rule engines**: eg Nexpert Object
- **Expert System shells**: typically goal-driven
- **AI languages**: eg LISP, PROLOG
- **AI / KBS development tools**: eg ART, KEE
- **Semantic web & ontologies**

Timeline:
- **1970s**: AI languages eg LISP, PROLOG
- **1980s**: Expert System shells typically goal-driven
- **1990s**: AI / KBS development tools eg ART, KEE
- **2000s**: Component-based Rule engines eg Blaze Advisor, JESS, JRules, Aion
How are Rule Engines evolving?

- **Rule engine** types:
  - Declarative (expressiveness) vs sequential (performance)
  - Forward & backward chaining and event-driven reasoning
  - Handling multiple rule types

- Rule execution **platforms**:
  - Java vs .NET vs C/C++ vs COBOL …
  - Embedded device ➔ PC ➔ web services ➔ mainframe

- Rule and policy **abstractions**:
  - Decision tables
  - Decision trees
  - Scorecards and score models
How are Rule Engines evolving?

- High **scalability & performance**:
  - Advances in rule execution algorithms
  - Rule servers for high multiples of transactions per second + simultaneous users
- Rule **expressiveness** to business users:
  - Rule syntaxes designed to be near Natural Language
- Multiple **data interface** capabilities:
  - 3GL (Javabean) support
  - Database and SQL support
  - XML support
  - Messaging and middleware support (CORBA, J2EE, MQ…)
How are Rule Engines evolving?

- Standards:
  - From 2002: **OMG** Business Rules Working Group now Business Enterprise Integration task force
  - Semantics for Business Vocabulary and Rules (est 2005) for formal business statement expressions
  - Production Rule Representation (est late 2005) for if.. Then.. Rule interchange across software models
  - Now: **W3C**: Rule Interchange studies underway
    - **RuleML** standards body is often associated with W3C
  - **OASIS**: BPEL: scope for adoption of OMG/W3C standard for rule interchange
  - **JSR**: JSR-94 Rule Engine Invocation standard for Java community

Rule tools are maturing fast
What is Rule Management?

- Rule engines only provide an alternative mechanism for implementing behavioral rules in software
  - Separation of rules = conforms to Business Rules Approach
- The main benefit from using a rule engine is the associated rule management process
  - Rule repository
  - Rule reporting, verification, validation
  - Rule metadata, versioning
  - Rule organization by business function

1990s Problem: rule representation & execution

2000s Problem: rule lifecycle management
“Business Rule Management”

what it means for IT
How does it all fit into applications?

**Business policy owners**
- Managers
- Business analysts

**Rule Authoring**

**Rule Repository**

**Rule Service**

**Business Application**
- Customers
- Employees
- Partners
- Suppliers

**Rule management software**

**Rule**: IF the Transaction is more than $1,000 and the Media is WEB THEN Require email confirmation
Software Best Practices

1. Develop Software iteratively
2. Continuously verify software quality
3. Control changes to software
4. Manage Requirements
5. Use Component-based architectures
6. Visually model software

How does Rule Management fit?
Crossover of OO and Rules

**OO Development**
- Defines business object model and infrastructure code
- Modeled in UML, developed by IT
- Subject to many software engineering best practices

**Rule Management**
- Declarative rules
- Defined in and outside of IT
- Organized into services, and rulesets
- Dependent on business object model
S/W Best Practices (1)

1. Develop Software iteratively
   - Implication: break down s/w development into manageable & measurable “chunks”
   - Declarative business rules are individual, testable units of decision making
   - BRMS allows the incremental development / test of business rules, separate from the other s/w parts

2. Manage Requirements
3. Use Component-based architectures
4. Visually model software
5. Continuously verify software quality
6. Control changes to software
S/W Best Practices (2)

1. Develop Software iteratively

2. Continuously verify software quality
   - Software approval / walk-through / test cycles
   - Business rules in near-English are easier to check
   - BRMS encourages constant rule validation and verification of RAD, XP

3. Control changes to software
4. Manage Requirements
5. Use Component-based architectures
6. Visually model software
S/W Best Practices (3)

1. Develop Software iteratively
2. Continuously verify software quality

3. Control changes to software
   - Software systems are “brittle” – small changes can break them
   - … yet “change” is guaranteed and must be managed
   - Business rules as declarative logic statements are designed for change
   - BRMS include higher-level rule maintenance facilities (eg Rule Maintenance Applications, Business Languages …)

4. Manage Requirements
5. Use Component-based architectures
6. Visually model software
S/W Best Practices (4)

1. Develop Software iteratively
2. Continuously verify software quality
3. Control changes to software

4. Manage Requirements
   - Requirements = use cases + descriptions, behavior
     see “Use Cases – Requirements in Context”
   - Business view:
     Requirements = business rules + system / process needs
   - BRMS provide a means of managing the rules in
     “requirements” in the form of explicit business rules,
     throughout the application lifecycle

5. Use Component-based architectures
6. Visually model software
S/W Best Practices (5)

1. Develop Software iteratively
2. Continuously verify software quality
3. Control changes to software
4. Manage Requirements
5. Use Component-based architectures
   - Divide-and-conquer approach to software development + re-use technical components off-the-shelf (eg a RDBMS for data management)
   - Separation of business rules from data, UI, middleware etc into its own component makes sense
   - BRMS include various rule execution services supporting different platforms and architectures
6. Visually model software
S/W Best Practices (6)

1. Develop Software iteratively
2. Continuously verify software quality
3. Control changes to software
4. Manage Requirements
5. Use Component-based architectures

6. Visually model software
   - “A picture is worth a 1000 words” – visualization is necessary for good communications (eg UML class diagrams)
   - Business rules and their interrelationships, cross-references to object models, and control flow should also be visualized
   - BRMSs provides powerful UI tools for modeling and viewing rules and their interactions with data
S/W Best Practices vs BRE and BRM

▶ Summary: a fit!
Business “Rule Management” vs “Process Management”

what are the differences?
Processes vs Rules

- Processes are higher level than rules
  - Rules can be used to implement processes
  - Rules are usually exposed as services: processes consume services
BPM vs BRM

- BPM is a w-i-d-e term
  - Business process modeling and simulation
  - Business process orchestration and flows
  - Business process automation and manual workflow
  - Business Performance Monitoring 😊

- Focus on Process definition and execution aspects

- BRM is much the same
Rules Automate and Reduce BPM Tasks

- New Hire
- Incomplete
- Withdrawn
- HR
- Allocate Employee Number
- Ready
- Wait for other tasks
- Deadline Warning
- Day before first day
- Induction Completed
- HR Assistant
- Inducted
- Unsignator
- Resubmit
- Withdrawn
- New Hire
- Review Hire Records
- Wait for first day
- New Employee Processing
- Employee started
- Supervisor
- Acknowledged
- Employee started
BRMs Can Include BPM Processes

Rules initiate BPM processes
Convergence?

- TBA
Case Studies
Onsite customer processing

DMV Offices

AIX Server

Rule Server

Local Rules Repository

Batch processing

Headquarters

OS/390 Mainframe

Rule Server

Local Rules Repository

New or updated rules are deployed from the master repository to each site once the rules are officially approved.

Rule design + build + test

Business Analyst

© 2005 Fair Isaac Corporation. All rights reserved.
Tax Board Architecture

Branch
- External App
- Phone
- Fat Client
- Thin Client

AIX Servers
- MQ Series Servers
- Websphere Application Server
- Web Server
- DB2 Taxpayer records

HQ
- Rule Server
- Rule Server
- Rules Repository

Tax Staff

Business Analyst
- Rule design + build + test
Telco Call Centre Architecture

Corporate

- Manager Viewpoint on rules

- Business Analysts

Call Center

- Deployment Manager
- Controller
- Backup Rules
- Profile DB
- Rule Server

Incoming Call Requests

Call Centre
Trade Validation & Compliance Architecture

Production

- Web Server
- BEA Weblogic J2EE
- Oracle
- Rule Server
- Customer & Trade Data
- Access

Development

- Deployment Manager
- Business Analyst
- Trading Rules
- Rule design + build + test
- Rule requirements from trader SMEs + regulators

Brokers & Traders

Oracle

Development

Rule requirements from trader SMEs + regulators
Q&A

References:
www.brcommunity.com
www.businessrulesgroup.org

Leaders:
www.brsolutions.com & www.kpiusa.com

Events:

Your presenter: paulvincent@fairisaac.com  www.fairisaac.com/rules