Towards enabling SaaS for Business Rules

Emilian Pascalau¹ & Adrian Giurca²

¹Hasso Plattner Institute
University of Potsdam

²Institute of Informatics
Brandenburg University of Technology

The International Workshop on Intelligent Service Management
March 24th, 2009, Leipzig, Germany
Agenda

- Motivation
- The Registry Architecture
- The Client-Registry Interaction
Motivation

- Rules are becoming increasingly important in business modeling and requirements engineering.

- The actual trend concerning aggregate enterprise that follows the Software as a Service (SaaS) paradigm envisions a continuous growth.

- To be able to use rules in SaaS based applications one has to be able to find and access rules in a service oriented manner. This presentation is about this.
The Requirements

- "service-based model of software is one in which services are configured to meet a specific set of requirements at a point in time" (Bennett et al, 2000)

- To be capable to change software easily to meet evolving business requirements.

- To be capable to issue semantic searches.

- Flexible interoperability opportunities with other systems and engines
The Ruleset Entry

A ruleset specifies the following information:

- An URI reference to the specific ruleset *implementation*, encoded by using `rulesetID` property;
- An URI reference to the ruleset *representation language*, encoded by using Dublin Core `dc:type`;
- A literal representing the code of the ruleset *addressed business*. It is a code (e.g. Naics, UNSPSC) of the corresponding business part (`dc:related`);
- An URI reference to the ruleset *format* of the representation (`dc:format`);
- An URI reference to the specific vocabulary implementation, expressed by using `vocabularyID` property;
- An URI reference to the vocabulary representation language (encoded with the Dublin Core `dc:type`);
- An URI reference to the format of the vocabulary representation (`dc:format`);
Registry Architecture

Why JBoss?
- JEE5 compliant; open source; one of the most used application servers

Why SPARQL?
- The registry entry content is semantic and SPARQL is the most used language for dealing with semantic queries

EntityManager (SPARQL-based)
- A sophisticated DAO manager. SQL based
- Here since the Database is XML/RDF, this should be SPARQL
Interaction Description

- Interaction is described with BPMN
  - BPMN is de facto the leading notation for business processes and complex interactions.
  - “a business process model consists of a set of activity models and execution constraints between them” (Weske, 2007)

- Four main processes:
  - Publishing
  - Updating
  - Deleting
  - Querying
The Publish Process

Client
- RuleSetEntry Publish Request
  - ruleSetEntry has been published
  - request has timed out
  - cancel RuleSetEntry Publish Request
  - cancel

RuleRegistry
- normal flow of publishing
- the user canceled the request
- handle Cancel Publish Sub-Process
- Cancel Publish Sub-Process

Publish Sub-Process
- verify Submitted RuleSetEntry
- save Submitted RuleSetEntry
- notify Successful Publish
- notify Timeout
- Cancel Publish Sub-Process

Process is ended by one of the activities or cancel events that enter in the gateway

Process is ended by one of the activities that enter in the gateway
The Update Process

Client

- update RuleSetEntry Request
  - normalFlow
  - not allowed to update
- allowed to update

RuleRegistry

- find ruleSetEntry
  - notFound
  - invalid credentials
  - valid credentials
- verify ClientCredentials
  - notAllowed

Update Sub-Process

- verify Submitted RuleSetEntry
  - update Submitted RuleSetEntry
  - notify Successful Update
- handle Cancel Update Sub-Process
- Cancel Publish Sub-Process

Process is ended by any of the activities that enter the gateway

- successfully Updated
- acknowledge Success
- operation timed out
- acknowledge timeout
- cancel Update
- cancel Rule Update
The Delete Process

Client

1. Delete RuleSetEntry Request
2. RuleRegistry
   - find ruleSetEntry
   - verify ClientCredential
   - allowed to delete

RuleRegistry

- notify ruleSetEntryId notFound
- notify notAllowed
- allowed to delete, request confirmation
- user cancelled deletion
- receive delete confirmation
- delete RuleSetEntry
- acknowledge Success Deletion
The Query Process

Client

- query Request
  - invalid Request -> acknowledge InvalidRequest
  - valid Request
    - no entry found based on request
      - new query Request
        - invalid Request
          - notify InvalidRequest
          - terminate process
        - valid Request
          - search RuleSetEntry
            - invalid query content
              - notify InvalidRequest
            - valid query content, continue with search
              - entry found
                - notify Found
                - acknowledge Found
              - no entry found based on query
                - notify NotFound
                - acknowledge NotFound

RuleRegistry

- verify queryContent
  - invalid query content
    - notify InvalidRequest
  - valid query content, continue with search
    - search RuleSetEntry
      - invalid Request
        - notify InvalidRequest
      - valid Request
        - entry found
          - notify Found
          - acknowledge Found
Thank you!