Towards enabling SaaS for Business Rules

Emilian Pascalau[1], Adrian Giurca[2]

[1]Hasso Plattner Institute
Prof.-Dr.-Helmert-Str. 2-3
14482 Potsdam, Germany
emilian.pascalau@hpi.uni-potsdam.de

[2]Brandenburg University of Technology
Walther-Pauer-Str. 3
03046 Cottbus, Germany
giurca@tu-cottbus.de

Abstract: The actual trends concerning enterprise development take heavily into account the Software as a Service paradigm. Business rules are widely used to define business behavior. Therefore, to be able to access business rules in this context a proper way of discovery and invocation is required. This paper describes the *architecture*, and *business processes* of a Semantic Web Service based registry for business rules.

1 Introduction and Motivation

The problematic of discovering and differentiating data is certainly not new in the software community. In order for data to be meaningful, the context of that data must be understood. As such, service discovery infrastructures are essential. Providing the ability to structure and model data and metadata to solve this problem is at the heart of any registry design, so as to prevent data from being undiscoverable or misunderstood. Nowadays, business rules are widely used in software applications. In the last 10 years business rules were employed to declaratively describe policies, business processes and practices inside enterprise. Rules are becoming increasingly important in business modeling and requirements engineering, as well as in Semantic Web Applications. There are several rule platforms and rule languages notably Jess, FLORA 2, Drools, Jena Rules, JSON Rules [GP08] but there is no standard way for defining business rules. The standardization is a concern for both OMG and W3C. The first produces OMG PRR Proposal[OMG07] and the second RIF-BLD [BK08]. The actual trend concerning aggregate enterprise that follows the Software as a Service [BLB+00] paradigm envisions a continuous growth (Market Trends: Software as a Service, Worldwide, 2007-2012, Gartner). An obvious necessity and requirement in order to be able to use rules in SaaS based applications is to able to find and access rules in a service oriented manner. Besides the already well known registries and repositories such as UDDI [OAS04] and ebXML [OAS01b, OAS01a] that enable enterprises with the ability to conduct businesses based on standards for business process collaboration, core data components, collaboration protocol agreements, messaging, there are also other types of registries such as metadata registries: NSDL Registry, Metadata Project. The founding principles of metadata registries were introduced in [HW02]. The authors argue in [HW02] that the approach of declaring schemas in metadata registries advance the W3C vision of

enabling a 'cooperative' Web where machines and humans can exchange electronic content that has clear-cut, unambiguous meaning, by providing a common approach for the discovery, understanding, and exchange of semantics. Therefore a registry to describe and discover business rulesets remains actual and necessary. First steps towards a business rules registry have been already done. While [GDPW08] introduced the entry information model for a business rules registry, this paper introduces the architecture of such a registry foreseeing also the SaaS demand. Besides the architecture also the Client/Registry interaction is depicted with the help of BPMN [OMG08] models. Moreover the registry will facilitate discovering of business rules by using Semantic Web techniques.

2 The Registry Architecture

There is a need of being capable to change software easily to meet evolving business requirements. More and more nowadays software development requires a shift towards a demand-centric orientation. This trail foresees the point were software will be delivered as a service within the framework of an open marketplace.

As stated in [BLB+00] "*the term software as a service is beginning to gain acceptance in the market-place; however the notion of service-based software extends beyond these emerging concepts*".

In such an approach a service conforms with the much accepted definition stating that a service is "*an act or performance offered by one party to another. Although the process may be tied to a physical product, the performance is essentially intangible and does not normally result in ownership of any of the factors of production*" [LVL96].

The authors in [BLB+00] argued that the "*service-based model of software is one in which services are configured to meet a specific set of requirements at a point in time*". Components may be bound instantly, just at the time they are needed - and then the binding may be discarded.

Based on these facts service based architecture is the normal consequence.

2.1 The Information Model

One proposal towards a ruleset entry model has been already introduced in [GDPW08]. However, we are going to use a simplified version of it. Therefore, the following information is required for a *web-based rule registry* entry:

Information related to the ruleset

1. An URI reference to the specific ruleset implementation, encoded by using *rulesetID* property;

2. An URI reference to the ruleset representation language, encoded by using Dublin Core `dc:type`;

3. A literal representing the code of the business addressed by this ruleset. It is a code (e.g. Naics, UNSPSC) of the corresponding business part (`dc:related`);

4. An URI reference to the format of the ruleset representation (`dc:format`);

Information related to the ruleset vocabulary

1. An URI reference to the specific vocabulary implementation, expressed by using *vocabularyID* property;

2. An URI reference to the vocabulary representation language (encoded with the Dublin Core `dc:type`);

3. An URI reference to the format of the vocabulary representation (`dc:format`);

All the above parts are required, and represent the minimal request of representing a rule set entry into the registry.

In addition to the above required information, users can provide optional information such as: (a) An URI reference to the natural description of the ruleset, (b) metadata of the ruleset creator, (c) the release date and others as they were discussed in [GDPW08].

As already been argued at the beginning of this section, the registry architecture is based on Web Services (See also [CMRW07, GHM+07] for more arguments on such solution).

2.2 The Architecture

The registry general architecture[1] is depicted in Figure 1 and at the first look follows the general principles of enterprise architecture. One notable distinction is that the Data Access Object Layer is based on SPARQL and not SQL. This is due to the fact that the Platform Specific Language of our registry is RDF.

An instantiation of the general architecture depicted in Figure 1 into an implementation specific one is illustrated in Figure 2.

The presentation layer from the general architecture (Figure 1) is represented in the instantiated architecture view (Figure 2) by two possible representation technologies JSP and Web Service. Same the business layer of the general architecture comprises two possible representations in the instantiation architecture: EJB and/or Web Service annotated EJB.

[1] Modeled with the Fundamental Modeling Concepts by using the ORYX browser based editor. The ORYX editor is a web-based editor (developed at the BPT chair, Hasso Plattner Institute) for modeling business processes in BPMN.

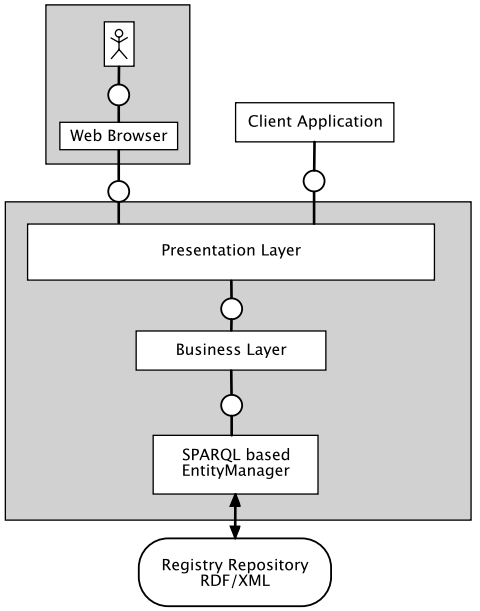

Figure 1: Registry's General Architecture

The instantiation architecture we propose is based upon JEE5 technologies and particularly EJB3 [Mic07]. As deployment environment we experiment with JBoss Application Server but any JEE5 compliant application server can be used as well. JBoss AS meets all of our platform requirements particularly it is JEE5 compliant, is open source, and is one of the most used application servers.

Recall that registry entries are encoded by using RDF based technologies. In addition, since the amount of data is likely to be low space consuming, our solution proposes to store data into RDF/XML files.

This architecture provides the user with a wide range of connectivity possibilities including: (a) either through a JSP web based interface easy to use for human users and (b) for machine clients in two flavors: (b1) SOAP Web Service based and (b2) a programming API. The JSP interface and the SOAP one are comprised inside the Web Server (Tomcat). The programming API is likely to be compatible with the JAXR[Mic02] Sun specifications. The specification provides a uniform and standard Java API for accessing different kinds of XML Registries. Also the API based approach could be useful for those applications that don't support Web Services.

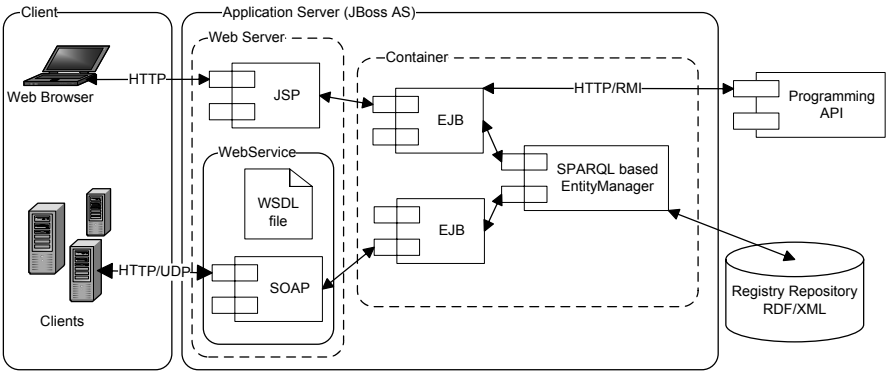

Figure 2: The Instantiation of the Registry's General Architecture

2.3 The Implementation

The web service is implemented as an annotated EJB3 and also the JSP interface communicates through EJBs by including SEAM framework. An important component is the *SPARQL based* `EntityManager` [Mic07]. The EJB3 technology comes with an already DAO layer in the form of the `EntityManager`. However, this EntityManager is SQL based and is suited for relational databases and not for RDF/XML repositories as in our case. Our approach is based on a Semantic Query Language SPARQL which is suited for our RDF/XML repository. SPARQL is a W3C standard and its purpose is to query, in principle the Web, but in general data models following semantic connections that exists between the models. The implementation of such an `EntityManager` will also enrich the JBoss Application Server with a new semantic query language.

3 Client-Registry Interaction

The Business Process Management Initiative (BPMI) has developed a standard Business Process Modeling Notation (BPMN) [OMG08]. The primary goal of BPMN is to provide a notation that is readily understandable by all business users such that business processes can be illustrated using a standard notation understandable also to business analysts that create the initial drafts of the processes, continuing towards technical developers responsible for implementing the technology that will perform those processes, and finally going on, towards the business people who will manage and monitor those processes.

As defined in [Wes07] "*a business process model consists of a set of activity models and execution constraints between them*". In today's business scenarios, service composition to provide added-value products to the market seems to be general trend. This section illustrates with the help of BPMN models the interaction between registry and its clients.

The registry interacts with the client through four processes: *publish*, *update*, *delete* and *query*. Architectural speaking there can be different types of clients (see Figure 2). However the registry itself interacts with these different types of clients based on the four operation types, using *the same workflow*. All processes models depicted in the following subsections comprise the `Client` pool and the `RuleRegistry` pool.

3.1 RuleSet Publish Process

The rule publish process is depicted in Figure 3. The process of publishing a `RuleSet` is started by the `Client` with a `ruleSetEntryPublishRequest` activity. The `Rule Registry` publish process is triggered by the message event of `RuleSet` publishing from a client. The process continues with the `Publish Sub-Process`. The Publish Sub-Process must end within a predefined time frame otherwise a `Timeout` exception will be raised. In case of a Timeout the client is notified by the activity `notifyTimeout`, and the `RuleRegistry` process terminates. The `Publish Sub-Process` comprises two activities: `verifySubmittedRuleSetEntry` and `saveSubmittedRuleSet Entry`. If the `Publish Sub-Process` has successfully ended then the client is notified of the activity success, and the process ends.

While the `Publish Sub-Process` is active the user can cancel its request. This is depicted with the help of the `cancelRuleSetEntryPublishRequest` activity, in the `Client` pool.

In the `RuleRegistry` pool this is modeled with some complex constructs. First the `RuleRegistry` receives the cancellation message. The cancelation request is handled by the `handelCancelPublishSub-Process` activity, followed by the `Cancel PublishSub-Process` event. An event with the same name is also attached to the `Publish Sub-Process`. In fact this is one and the same event. This pattern is called *Cancel Case* according to [WvdAD+06]. Besides the *Cancel Case* pattern, this model takes use also of *Send Pattern* and *Receive Pattern*.

3.2 RuleSet Update Process

The process of updating a registry `RuleSetEntry` (See Figure 4) starts with a `Client updateRuleSetEntryRequest`. In this case the message sent to the `RuleRegistry` contains the `ruleSetEntryId` and the `clientCredentials`. The registry receives the request and tries to find the `ruleSetEntry` requested by the client. If the entry is not found then the registry informs the client and the update process is finished for both parties. If the entry is found then the registry validates client's credentials. Only the client that published the `ruleSetEntry` is allowed to update the registry entry. When these requirements are met then the registry sends the complete `ruleSetEntry` to the client. In such a way the client is able to update the entry's content and sends it back to the registry. As for the `publish` operation, the `update` operation is time based, so it has to

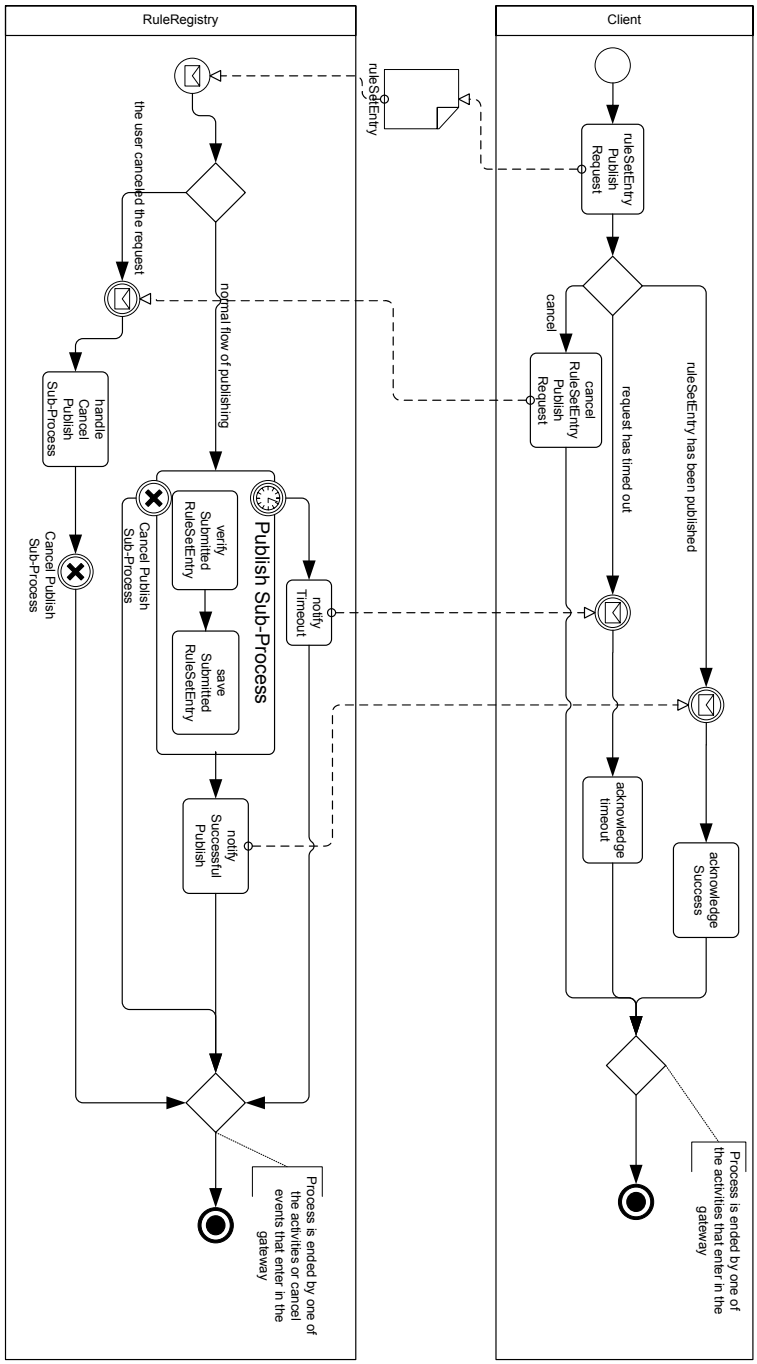

Figure 3: RuleSet Publish Process

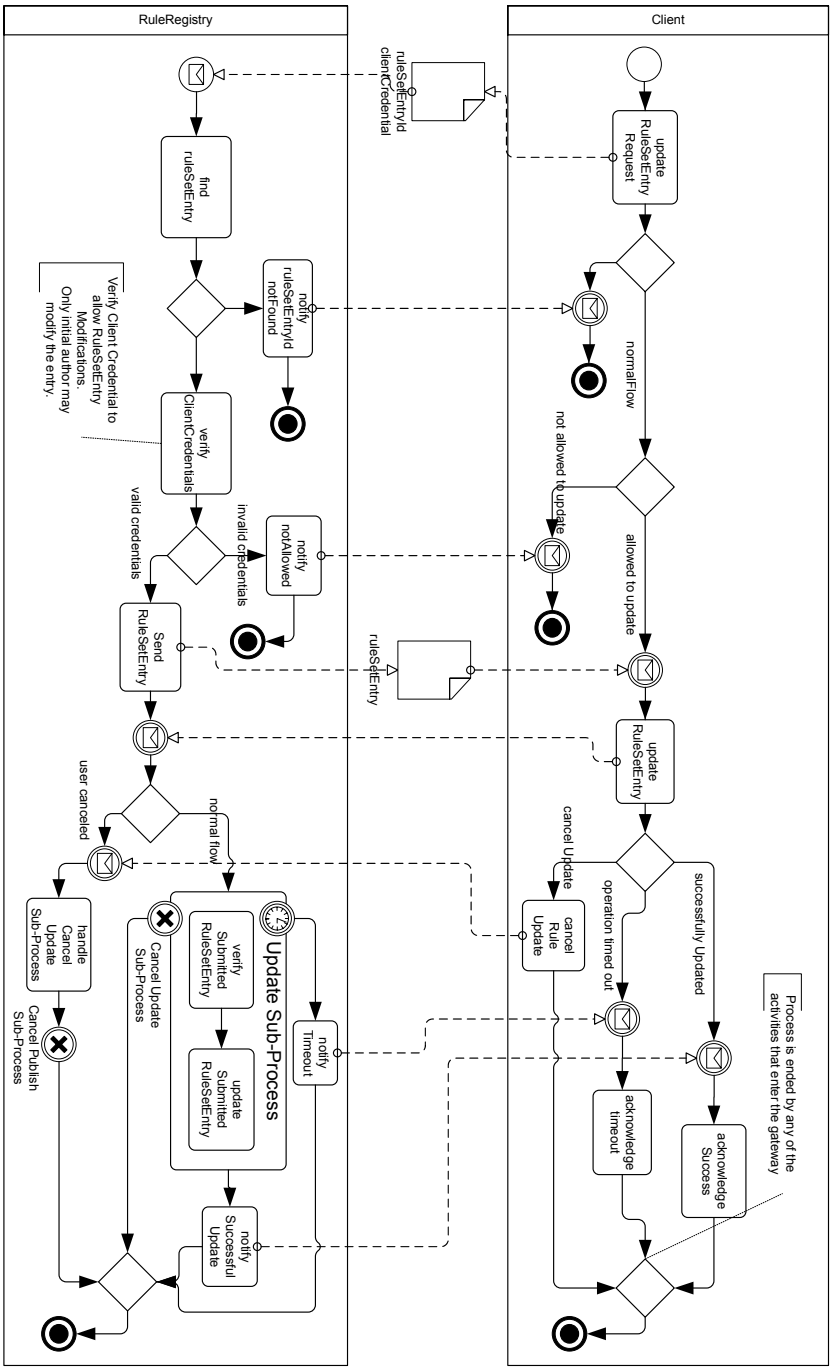

Figure 4: RuleSet Update Process

be fulfilled in a specific time. During this time the client is allowed to cancel the process.

3.3 RuleSet Delete Process

The `delete` process is illustrated in Figure 5 and is similar with the update process. What it brings new is that in this case the client is requested to confirm the operation. Again only the initial publisher is allowed to delete the `ruleSetEntry`. After delete confirmation is received the registry's process continues with atomic activities. No sub-processes are involved.

3.4 RuleSet Query Process

The `query` process (Figure 6) compared to the other three processes already discussed looks as being the simplest. The `Client` requests a query by sending a `QueryRequest` message to the `RuleRegistry`. This message is the starting event for the `RuleRegistry` query process. The registry verifies the query's content. If query's content is invalid then informs the `Client` about the invalidity of the request and ends its own process. Receiving a message stating that the initial request was invalid, makes the `Client` able to choose from two flows: either acknowledging the invalidity of the request and then terminating the process or acknowledging the invalidity of the request but issuing a new query towards the registry.

If the request is valid then the registry searches for the `RuleSetEntry` fulfilling the query content and, if a `RuleSetEntry` is found then the registry forwards it towards the `Client`. In case that no `RuleSetEntry` has been found the `Client` can either issue another query or terminate its process.

This process takes advantage of the *Arbitrary cycles* pattern described in detail in [Wes07, WvdAD+06]. A small excerpt of Figure 6, depicting only the loop is presented in Figure 7.

4 Conclusion and Future Work

This paper proposes architecture for a business rule registry. The architecture is service based and foresees the general actual trend of developing new software in the context of software as a service (SaaS). One core part is the BPMN modeling of registry business processes. Such approach is well suited for both business administration professionals and also for software engineers and software architects that must provide the concrete implementation. Future work envisions concrete implementation, and how the registry comes in handy in real life business use cases. Besides all, the need for providing registry services also in REST based style has to be investigated.

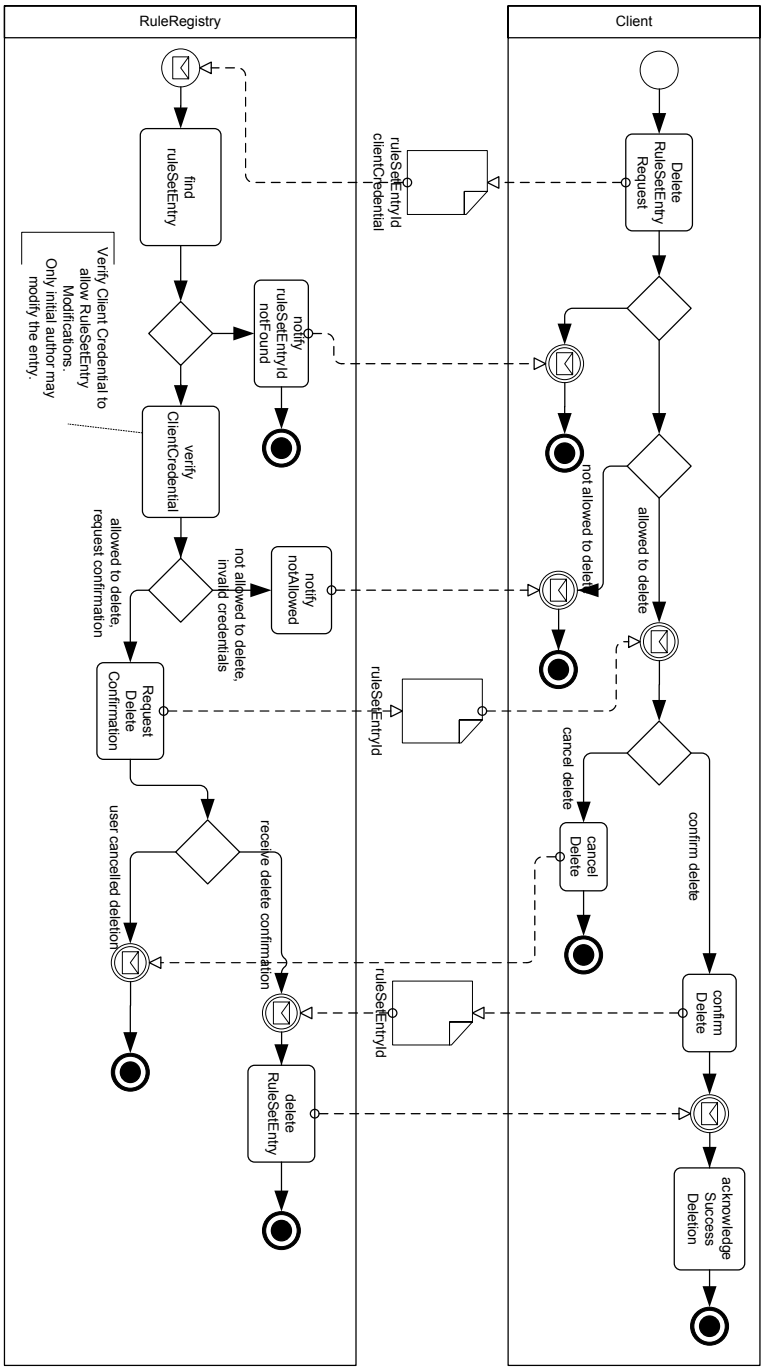

Figure 5: RuleSet Delete Process

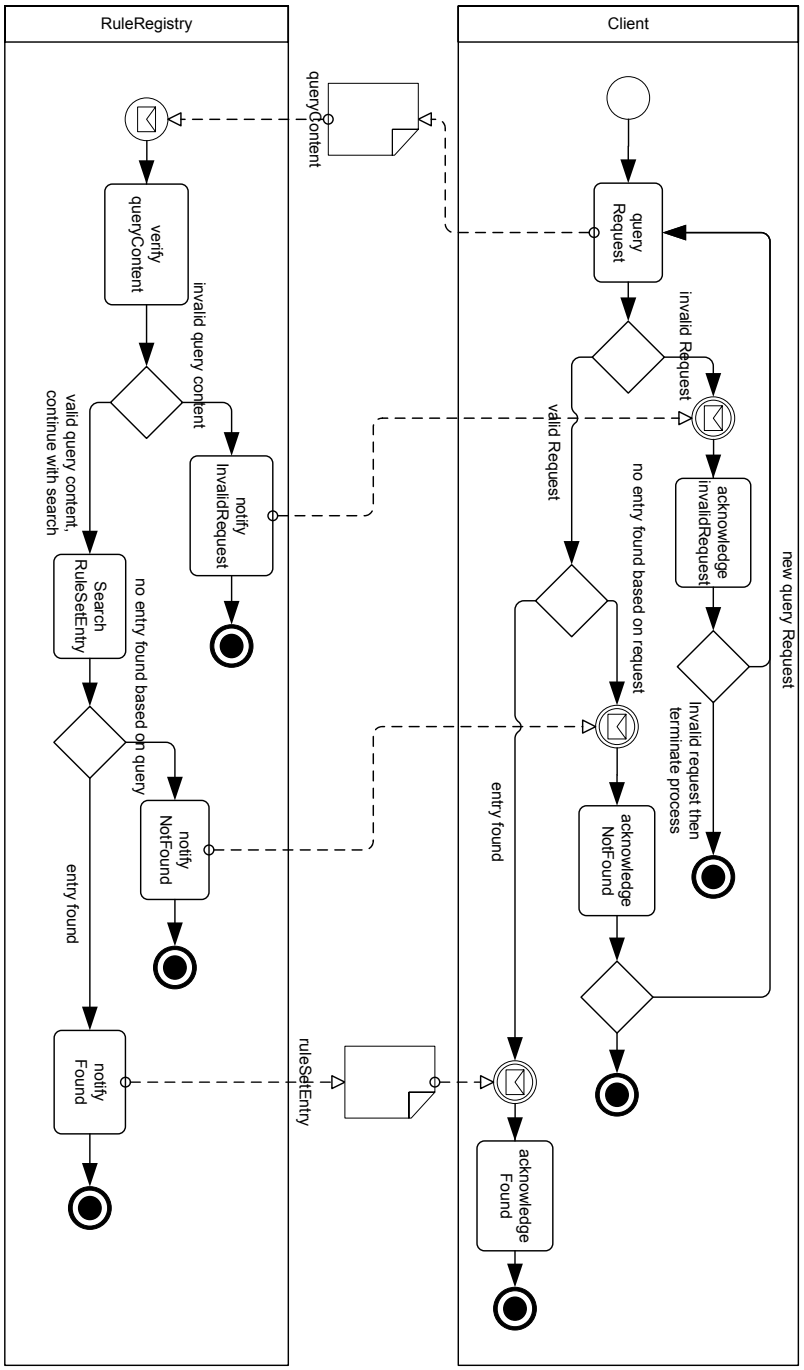

Figure 6: Query Process

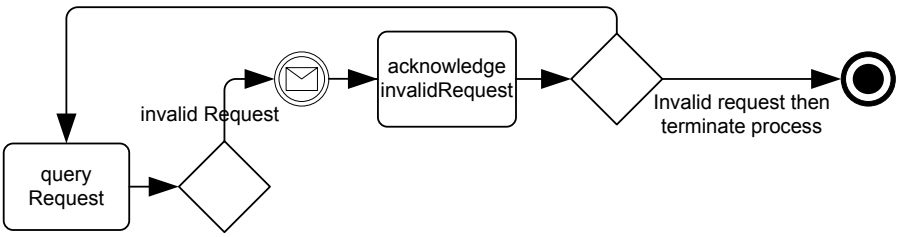

Figure 7: Loop

References

[BK08] H. Boley and M. Kifer. RIF Basic Logic Dialect. http://www.w3.org/2005/rules/wiki/BLD, 2008.

[BLB+00] K. Bennett, P. Layzell, D. Budgen, P. Brereton, L. Macaulay, and M. Munro. Service-Based Software: The Future for Flexible Software. In *Proceedings of the Seventh Asia-Pacific Software Engineering Conference (APSEC2000)*, pages 214 – 221. IEEE Computer Society, 2000. http://www.bds.ie/Pdf/ServiceOriented1.pdf.

[CMRW07] R. Chinnici, J. Moreau, A. Ryman, and S. Weerawarana. *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*. W3C, 2 edition, June 2007. http://www.w3.org/TR/wsdl20/, retrieved Novemeber 2008.

[GDPW08] A. Giurca, I. Diaconescu, E. Pascalau, and G. Wagner. On the Foundations of Web-based Registries for Business Rules. In *Proceedings of the 2nd International Symposium on Intelligent Distributed Computing, IDC2008*, volume 162 of *Studies in Computational Intelligence*, pages 251 – 255. Springer Berlin / Heidelberg, 2008. http://dx.doi.org/10.1007/978-3-540-85257-5_26.

[GHM+07] M. Gudgin, M. Hadley, N. Mendelsohn, J. Moreau, H. Frystyk Nielsen, A. Karmarkar, and Y. Lafon. *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*. W3C, 2 edition, April 2007. http://www.w3.org/TR/soap12/, retrieved Novemeber 2008.

[GP08] A. Giurca and E. Pascalau. JSON Rules. In *Proceedings of the Proceedings of 4th Knowledge Engineering and Software Engineering, KESE 2008, collocated with KI 2008*. CEUR Workshop Proceedings, 2008.

[HW02] R. Heery and H. Wagner. A Metadata Registry for the Semantic Web. *D-Lib Magazine*, 8(5), May 2002. http://dlib.org/dlib/may02/wagner/05wagner.html.

[LVL96] C. Lovelock, S. Vandermerwe, and B. Lewis. *Services Marketing* . Prentice Hall Europe, 1996.

[Mic02] Sun Microsystems. JSR 93: JavaTM API for XML Registries 1.0 (JAXR). http://www.jcp.org/en/jsr/detail?id=93, June 2002.

[Mic07] Sun Microsystems. JSR 220: Enterprise JavaBeansTM 3.0. http://jcp.org/en/jsr/detail?id=220, November 2007.

[OAS01a] OASIS. ebXML Business Process Specification Schema Version 1.01. http://www.ebxml.org/specs/ebBPSS.pdf, May 2001.

[OAS01b] OASIS. ebXML Technical Architecture Specification v1.0.4. http://www.ebxml.org/specs/ebTA.pdf, February 2001.

[OAS04] OASIS. UDDI Version 3.0.2, UDDI Spec Technical Committee Draft, Dated 20041019. http://uddi.org/pubs/uddi-v3.0.2-20041019.pdf, October 2004.

[OMG07] OMG. Production Rule Representation (PRR), Beta 1. Technical report, OMG, November 2007.

[OMG08] OMG. Business Process Modeling Notation, V1.1. http://www.omg.org/spec/BPMN/1.1/PDF, January 2008.

[Wes07] M. Weske. *Business Process Management: Concepts, Languages, Architectures* . Springer-Verlag Berlin Heidelberg, 2007.

[WvdAD+06] P. Wohed, W.M.P. van der Aalst, M. Dumas, A.H.M. ter Hofstede, and N. Russell. On the Suitability of BPMN for Business Process Modelling . In *Business Process Management*, volume 4102 of *Lecture Notes in Computer Science*, pages 161–176. Springer Berlin / Heidelberg, 2006. http://dx.doi.org/10.1007/11841760_12, http://www.workflowpatterns.com/documentation/documents/BPMN-eval-BPM06.pdf.