A Business Rules Driven Framework for Consumer-Provider Contracting of Web Services

Werner Mach
University of Vienna
Faculty of Computer Science
Vienna, Austria
werner.mach@univie.ac.at

Benedikt Pittl
benedikt.pittl@gmx.at

Erich Schikuta
University of Vienna
Faculty of Computer Science
Vienna, Austria
erich.schikuta@univie.ac.at

ABSTRACT

In this paper we present a scalable and extensible architecture of a business rule management framework. This representation can be used for agent based automatic negotiation and re-negotiation of web services. To ensure scalability and extensibility our architecture is based on the service oriented design pattern using ontologies. Finally we develop a prototype based on our business rules framework which simplifies business logic modification and maintenance for both service-provider and -consumer.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; H.3.5 [**Online Information Services**]: [Web-based services]

General Terms

Management, Economics, Design, Theory, Verification

Keywords

Business Rules, Workflow, Ontology, Blackboard, Semantic Web

1. INTRODUCTION

Over the last years the management of business rules gained strong attention as a novel approach to handle business logic in information systems. Business rules can be maintained by business users both reducing adaptation time and increasing flexibility. Currently, business logic is typically stored in software. Hence, changing business logic requires a software specialist who modifies the software code. Usually the software specialist, who implements the adaptation of business logic, doesn't know details about a companies business domain leading to misunderstandings and errors in turn. Therefore, storing business logic in software is considered as being error-prone and hard to maintain. Thus, mapping business logic into explicit business rules increases maintainability and usability as business users can use tools or languages to manage business rules directly, without support from software specialists.

In our previous research we introduced a generic framework supporting automated negotiation and re-negotiation of consumer-provider contracting of web services [12] based on an elaborated cost model [13]. This framework is designed as a self governing infrastructure. The core part is a knowledge base which enables the framework making decisions during negotiation or re-negotiation autonomously, i.e. without any interaction of humans. Both provider and consumer have access to a knowledge base which consists of both a business rules repository implementing the business strategy and an economic cost model allowing evaluation of business decisions. For justification we implemented a prototype of a scalable and extensible workflow based on business rules.

The layout of the paper is as follows: First we present the state of the art on business models and business rules in sction 2. We highlight the benefits of using business rules and their fields of application as well as their deduction from a business model. In section 3 we present an ontology for business rules allowing for the definition of arbitrary business strategies for a wide field of enterprises. Our proposed negotiation and re-negotiation framework is introduced in section 4 followed by an implemented example workflow fostering business rules in section 5. The paper is closed by a summary and sketch on future work.

2. STATE OF THE ART

In the following we give a short survey about business rules and enabling semantic web technologies for better understanding the rational for choosing the appropriate tools for our approach.

2.1 Business Rules

Many authors [18, 22, 26] cope with business rules. Unfortunately, each of these authors provides different definitions. Boyer and Mili [3] summarizes several definitions by outlining two characteristics of business rules.

- **Business rules are about business.**
  A company's business motivation is the driver for creating business rules. For example, this motivation can aim on maximizing profit or minimizing risks. Business rules are formulated to attain these aims [3]. Business rules may not only represent the organization's

iiWAS 2013, Vienna, Austria

business motivation but also legislation, regulations, external standards and best practices [26].

- **Business rules focus on structure and behavior.**
  Boyer and Mili [3] distinguish between two types of rules. Structure rules describe the structure of a business, e.g. *"A sale consists of a buyer, a product and a bill."*. Behavior rules describe how business reacts in respond to an event, e.g. *"If loan is requested then liabilities have to be checked."*

An insurance company will have rules coping with the insurance premium calculation whereas a bank will have rules for granting loans. A simple rule of an insurance company may be *"IF a customer has an income less than 1000$ THEN the credit enquiry is refused"*. This example shows, that business rules can be described by simple statements building upon *"IF-THEN"* structures. Each company has thousands of such simple business rules. However, they may not be documented explicitly or they are hidden in softwareWitt [26] emphasizes that the rules have to be accessible for all stakeholder who are influenced by the rules. Furthermore it must be possible to update the rules in an adequate time. Rules in natural language are easy to understand by the business users.

However, natural language rules may be ambiguous and the same rules can be formulated in many different ways. For clear natural language rules you have to constrain the vocabulary as well as the syntax. In order to standardize the format and the business terminology, the Object Management Group (OMG) published the Semantics of Business Vocabulary and Business Rules (SBVR) as a standard for defining business rules in natural language. A SBVR vocabulary consists of terms and facts, whereas the facts represent the relationship between terms. Based on defined terms and facts rules can be formulated complying to the SBVR by defined fact models. Fact models are used to standardise vocabulary by defining the terminology used within rules [26].

Business rules are executed within business processes by both IT-systems and humans [3]. Within business processes, business rules represent the decision logic [17]. In order to automate business processes by IT-systems, business rules have to be defined in a computer readable way. We distinguish between two type of business rules:
(i) **Operative Rules** define the actions triggered by an event, e.g.
R1: Each service must *specify* a start time and an end time.
R2: The service start time *specified* in each service must *be not earlier than* the service request time.
(ii) **Structural Rules** how various constructs are defined by the organization, e.g.
R3: A premium customer *is* defined as a customer whose total revenue *is greater than* 50000$ [1] in the last 5 years.
R4: A premium discount *is* by definition exactly 20 percentage of each service price.

2.2 Business Processes

A business process describes a cross departmental series of steps to produce valuable output for the customer. A process could, for example, describe the handling of a customer order within a company [19]. In the following sections we provide a description of the anatomy of a process. As [15] shows, each process has five components:

- **Trigger.** A trigger kick-starts a process. The trigger of the order process could be a customer placing an order.
- **Input.** Inputs are resources needed for executing the process. This could be for example an order document.
- **Enabler.** Enablers represent the necessary equipment for the process execution like a software system for processing an order.
- **Guide.** A guide points out how to execute a process.
- **Output.** As processes aim on customer satisfaction the result of an process is something valuable for the customer.

As already mentioned, business processes are executed by both IT systems and humans. In order to automatize business processes by IT systems the process has to be transformed into a workflow, i.e. a workflow describes how to execute a process. Hence, the business logic, which can be represented by business rules, has to be accessible for IT systems. Usually this is done by mapping the business logic into software code. Unfortunately business users who are responsible for maintaining and creating business rules are not able to handle business rules stored in software. Thus, they have to communicate with IT specialists who implements the rules in software code. Not only the creation of new rules but also the change requests of already existing rules require the skills of an IT specialist. Typically, the IT specialist may not know details about a rule's business domain whereas the business user may not know anything about software development. Hence misunderstandings can occur and consequently this approach can be considered as being error prone [4]. Further problems result that rules are hidden in several code paths instead of a central repository [4]. Rules stored this way are hard to manage and violate many principles [18]: Rules should

- ... be managed directly by the people who have the essential knowledge.
- ... be put down in writing and published.
- ... be written in natural language.
- ... be independent of workflows.
- ... be based upon facts. The facts should be based on concepts.
- ... control the behaviour.
- ... be driven by the important business factors.
- ... be manageable by authorized persons.
- ... be handled from a single source.
- ... be administrated.

Rules managed by Business Rules Management Systems are able to comply to these principles.

[1] The SBVR convention is to mark quantities, dates and times using a double underline

Table 1: Simplified decision table

Storage capacity (Tera Byte)	Discount (%)
1,3	0
4,7	2
8,10	5

2.3 Business Rules Management Systems

The prominent principle published by [18] reinforces, that business user must be able to formulate the business rules on their own. Generally, there are four ways how business users can create business rules.

1. **Programming Language.** Business user can use UML or programming languages like Java to define business rules. However, usually business users don't know theses languages.
2. **New language.** Business users can write their rules in a novel language similar to a natural language which is readable by both, IT systems and humans.
3. **User interfaces.** Use interfaces can be used to support business people managing rules.
4. **Restrict natural language.** The natural language approach could be limited to set of vocabularies sufficient to create rules for a special business domain.

All proposed possibilities, besides the first one, require the definition of a vocabulary or ontology which can be processed and interpreted by both, humans and IT systems. An exact definition of a vocabulary or ontology prevents ambiguities of terms [9].

Business Rules Management Systems (BRMS) storing the business rules representing the business logic of a company in a repository and are assisting in the management of the rules by providing several tools and interfaces. This enables reusability, faster development, clearer auditability and consistency.Many companies offer Business Rules Management Systems.Two very popular BRMSs are Drools and IBM WebSphere ILOG JRules:

- **Drools.** Drools [7] is a comprehensive open source BRMS framework based on Java focusing on business logic integration. There are several ways to formulate rules in Drools. One method is to build decision tables in Microsoft Excel ®. Table 1 shows a simplified decision table for the pricing model of a cloud provider. The left column shows the storage capacity whereas the right column shows the discount. If a consumers buys for example 5 Terabyte, he gets a discount of 2%.

  Another way to define rules within the Drools framework is to write rules in the so called *mvel* dialect. The rules formulated in this language look similar to Java source code.

- **IBM WebSphere ILOG JRules.** IBM WebSphere ILOG JRules [3] is a comprehensive commercial BRMS platform. This platform allows to create rules in different ways too. There is a "rule studio" assisting the creation of decision tables. But there is also a mechanism to create rules by a restricted natural language.

2.4 Business Models

Business models have the goal to create value, build new businesses or improve existing businesses. There exists a broad range of business model definitions, e.g. by Alt and Zimmermann [1]: *"For the business model discussion [...] we will distinguish six generic elements of a business model: Mission, Structure, Process, Revenues, Legal Issues, Technology".* Gordijn [8] gives a definition which includes activities for creating values and covers the whole life cycle of products or services: *"We define e-Business models as conceptual models that show how a network of actors (a value constellation) creates, exchanges and consumes objects of value by performing value adding activities."* Osterwalder et. al. [16] give a more general and comprehensive definition of business models: *"A business model describes the rationale of how an organization creates, delivers, and captures value. The business model is like a blueprint for a strategy to be implemented through organizational structures, processes, and systems".*

For development of our framework we used Osterwalder's business model. We chose his definition of business models to show how business rules can be derived from a business model expressed in natural language. He describes a business model by means of nine basic building blocks that show the logic of how a company intends to make money. These nine blocks cover four main areas of a business: customers, offer, infrastructure and financial viability.

(i) **Customer Segments** are the most important parts of each business model. Different groups of customers should be handled in different ways. Organization and their processes must consider these groups. (ii) **Value Propositions** describe the specific products and services for a specific customer segment. (iii) **Channels** include the description how the company communicates to reach the customer segments. The purpose of the channels is to deliver value proposition and is the interface to the customer. (iv) **Customer Relationships** describe the companies type of relationship to each customer segment, e.g. personal or automated or a mixture of them. (v) **Revenue Streams** hold the representation of all cash streams to each customer segment. Different pricing mechanisms can be used for each revenue stream. Osterwalder distinguishes between two types of revenue streams: revenues resulting from one-time customer payments or from ongoing payments. (vi) **Key Resources** create value propositions in each company. The resources can be physical, financial, intellectual or human depending on the type of business. (vii) **Key Activities** should be described to make and keep the business model work, e.g. for a hardware manufacturer supply chain management is the key activity. (viii) **Key Partnerships** represent the description of the network of a company. These partnerships reach from a buyer-seller partnerships to strategic alliances. (ix) **Cost Structure** describes the most important cost during operation of the business model. That means all cost for creating, generating and maintaining value and customer relationships. This cost can be derived directly from the previous building blocks such as key resources, key activities and key partnerships.

3. A BUSINESS RULE REPOSITORY

In our work we follow the approach that the building blocks of a generic business model can be represented as on-

tology. Ontologies are used for structuring data. Maedche and Staab [14] describe ontologies as the following: *"Ontologies are (meta)data schemas, providing a controlled vocabulary of concepts, each with an explicitly defined and machine processable semantics."*

Chandrasekaran et. al. [5] distinguish between two related fields of applications of the term ontology in IT. First, an ontology is the definition of the vocabulary used for describing a specific domain. Additionally the relationships between the terms are defined. Second, a representation vocabulary used to describe knowledge is called ontology too. The representation vocabulary contains the vocabulary which is used to define facts and consequently describe knowledge of a specific domain. So, ontologies determine types of objects, which represent things in the real world, as well as their attributes and their relationships. This structure is used to represent knowledge.

3.1 An Ontology for Business Rules

Figure 1 illustrates our generic business model ontology as UML class diagram. Each business model consist of building blocks similar to Osterwalder's framework and constrain data. Our business model ontology contains 1..n building blocks. That means the resulting business model is not limited and can easily be extended or shrunk to fit the companies needs.

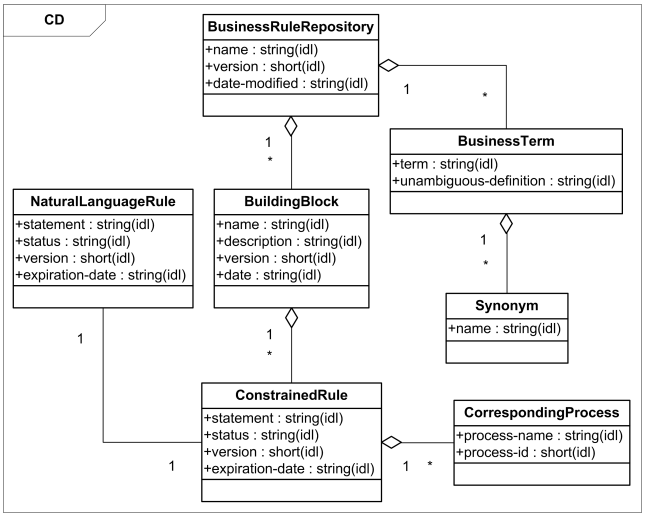

Figure 1: Generic Business Model Ontology

The ontology business rule repository consists of following elements: (i) **Business Rule Repository** is the root element and has the attribute *name* and *date-modified*. The business rule repository has 1..n building blocks, 1..m business rules expressed in natural language and 1..p business term definitions. (ii) **Building Block** holds the information about the core description and characteristics of the companies business. The attribute *description* is used for defining the building block. Each building block has 1..n associated constraint rules. The constraint rules are derived from the business rules described in natural language. (iii) **Natural Language Rule** defines the business rules expressed in natural language. It is important to have the attribute *version* and *date-modified* for supporting the history management of business rules. (iv) **Business Terms** describes each term used in the constrained rules. (v) **Synonyms** are only used for better understanding the defined business terms, so that all participants involved have a clear understanding about the meaning of each term. Each business term can have one or more synonyms. (vi) **Constrained Rule** defines the business rule derived from the natural language rule and is used in the processes instead of the natural language rule. (vii) **Corresponding Process** holds the information in which process a rule is being used. Each rule is used in one or more processes.

Using this generic ontology a company is able to build its own specific business rule repository. The process for generating a business rule repository is depicted in algorithm 1:

```
Data: Business Rules defined in Natural Language 1..R
Result: Constrained Business Model Repository
build constrains for the business model;
define appropriate building blocks 1..B;
for each business rule R do
    for each building block B do
        if business rule R is related to building block B
        then
            translate business rule R from natural
            language to constrained language;
            store translated rule in building block B;
            fill in meta-data for rule R
            (status,version,expiration date);
            break;
        end
    end
end
```

Algorithm 1: Generating Business Rule Repository

3.2 Realization by Semantic Web Tools

The W3C has published the XML based standards Ontological Web Language and the Resource Description Framework for defining ontologies.These two standards origin from the semantic web technology development. As these technologies are platform independent, exchangeable, comprehensive and widely-accepted we use these technologies to build our architecture of a workflow negotiation for business applications. For the management of the knowledge base we apply SPARQL [25] and Jena [10].

Summing up, RDF and OWL are used for ontologies, whereas SPARQL is used to query these ontologies, and Jena is used to execute rules from the knowledge base.

3.2.1 Resource Description Framework

The Resource Description Framework is a W3C standard for building ontologies [24]. Ontologies are knowledge bases which contain knowledge of a specific domain. In RDF, knowledge is represented by so called resources and their relationship to other resources. A resource represents an object of the real-world like a car or a book. RDF as well as OWL are technologies for creating ontologies.

3.2.2 Ontological Web Language

RDF can not be used to describe complex ontologies. Consequently the Ontological Web Language (OWL), which is based on RDF, was developed to build more complex ontologies. The W3C published three OWL standards which

are called OWL Full, OWL DL and OWL Lite. OWL Full is biggest standard which encompasses both, OWL DL and OWL Lite. OWL Lite is the smallest standard and is part of OWL Full and OWL DL [23].

3.2.3 SPARQL

RDF and OWL are used to store knowledge. SPARQL Protocol And RDF Query Language (SPARQL) is a language for querying this knowledge. Just as RDF and OWL, SPARQL was published as standard by the W3C [25].

3.2.4 Jena

Apache Jena is a Java based framework for building semantic web applications [10]. The framework provides an API for processing OWLs as well as RDFs and allows to query them via SPARQL. Additionally, Jena provides a rule-based inference engine which allows to execute Jena rules. Due to the fact that Jena is a core part of our approach we will describe the structure of Jena rules in more details.Listing 1 shows a rule in the turtle syntax. This rule is used to determine premium customers.

Listing 1: Jena Example Rule

```
@prefix j.0: http://www.univie.ac.at
[premium: (?s rdf:type j.0:PremiumCustomer)
          <-
          (?s rdf:type j.0:customer)
          (?s j.0:customer_Revenue ?c)
          greaterThan(?c,60)
]
```

1. First of all, prefix definitions can be declared before the rule is defined. A prefix is initiated by the *@prefix* keyword followed by the prefix name and the URI.

2. The rule itself is defined within squared brackets. The rule may start with the rule name. In listing 1 the rule is called *premium*.

3. Jena supports the forward, backward and promiscuous rule mode. Forward rules have the structure T_1, T_2, T_n ->T_0. T stands for triple. The structure of the forward rule can be interpreted as if T_1, T_2, T_n are matching, T_0 is executed. Backward rules have a similar structure like T_0 <- T_1, T_2, T_n. T_0 is executed if the triples T_1, T_2, T_n are matching. It is also possible to combine forward and backward rules which requires Jena's promiscuous mode. The rule in listing 1 is a backward rule as it contains the <- arrow.

4. The rule in listing 1 has three triples.
   T_0: ?s rdf:type j.0:PremiumCustomer
   T_1: ?s rdf:type j.0:customer
   T_2: ?s j.0:customer_Revenue ?c

   Within these triples, variables are used. In T_1, a customer is stored in variable ?s. In T_2, the revenue of the customer, which is stored in ?s, is loaded into the variable ?c. Finally this variable is used in a so called *built-in*. A *built-in* is similar to a procedure in a programming languages. In listing 1 the *greater-than built-in* is used. This greater-than checks if the revenue is greater than 60. In this case T_0 is executed which means that a customer becomes a premium customer. Each customer who complies this rule becomes a premium customer by getting the type "premium customer."

In the following section 4 we use the resulting business model repository in our automated negotiation and re-negotiation framework.

4. A NEGOTIATION AND RENEGOTIATION FRAMEWORK

Focus of our recent research was the definition of a novel negotiation and re-negotiation framework [12] which allows for automatic commerce (negotiation and contracting) of Web services based on economic principles. This approach enables market-based service trading (following a bazaar style) and extends the classical supermarket approach typical for service negotiation today. We extended the WS-Agreement standard by feasible workflows to support auctioning for negotiation and re-negotiation. By mapping of business strategies defined by economic goals of the respective organization into an ICT enabled framework, our framework facilitates autonomic agents acting as organizational representatives stipulating service level agreements without human interaction. This allows for business transactions transparently to the environment but adhering to business objectives of the originating organization (i.e. company, industry, community, etc.).

4.1 Knowledge Base Architecture

Figure 2 illustrates the knowledge base of the framework[2] introduced in [12]. The framework is build up on a symmetrical architecture as the consumer agent as well as the provider agent will have the same components:

- **Negotiation and re-negotiation engine.** The negotiation and re-negotiation engine is the main component of the framework. It is responsible for negotiation and re-negotiation phase and accesses the knowledge bases.

- **Knowledge base.** The knowledge base consists of three components. The business rules repository contains the business rules of an agent which represent the business strategy. The economic cost model stores the costs of each production factor like disc space, computational power and is responsible for determining an adequate price. Finally the history data component stores the historical data of the customer and provider agent. All these data sets provide the basis for the decision process during the negotiation phase.

- **Service template registry.** The service template registry contains the offered services of the providers. It can be used by the consumers to look for appropriate services.

- **Consumer- and Provider Agents.** The agents act as interfaces between the framework and applications or humans. It is used for managing services, modifying the knowledge base and for controlling the auctioning.

- **Auctioneer Agent.** The auctioneer agent is used to supervise the auctioning process.

- **Protocols.** We adhere to the principle of fostering existing standards. Therefore our framework is based

[2]Please note that the figure shows only parts of the framework

on already existing standards only, such as the WSDL, WS-Agreement, etc.

For implementation of our business rule environment we used basically two software components:

- **D2RQ platform.** The D2RQ platform is able to map MySQL, Oracle and other databases into a RDF structure. Basically, the platform encompasses the D2RQ engine and the D2RQ server. The engine is responsible for converting Jena API calls to SQL queries whereby the server allows to access the database via SPARQL queries. This data will be merged with other RDF files and processed by Apache Jena.

- **Apache Jena.** Apache Jena processes the RDF data including inferring Jena rules. The result of the inferring process is a RDF which can be queried by the SPARQL engine provided by Jena. The query result is sent back to the client application for further processing.

Figure 2 shows, that the customer's as well as the provider's knowledge base encompass the Jena framework. Data is queried from the database via the D2RQ platform and passed to Jena in order to inference business rules. In the following we describe a scalable and extensible workflow based upon the environment.

Our presentation comprises two steps: first, we define the goals which should be achieved by a workflow executed based upon the business rules environment. Then we present the architecture of this environment using the technologies we presented before.

The workflow we create should fulfill the following goals:

- **Business rules management.** The workflow makes decisions based on business rules instead of business logic stored in software code. This improves changeability and maintainability.

- **Flexibility.** The workflow can be extended and modified easily. It should be possible to add and drop tasks of the workflow.

- **Scalability.** The workflow should be able to handle big workloads by elastic resource management.

To create a flexible and scalable workflow we followed the Service Oriented Architecture principle. By sending requests to different services at same time, requests will be executed parallel, if they are deployed on different servers.

Regarding our goals, this architecture enables flexibility as you can consume the services you need in the workflow and replace them if you do not need them any more. Scalability can be achieved by horizontal and vertical scalability. Horizontal scalability means deploying the service several times on different servers. Vertical scalability can be achieved by adding additional resources to servers which host the heavy used services.

Our third goal is to handle the business logic in the services by business rules. Therefore we use several frameworks and servers within the services as shown in Figure 4 .

The following steps are executed by a service:

1. **A request is sent to the service.**

Table 2: Customer Database Table

ID	Name	Total [3]	Annual [4]	Premium
124533	Lewis	80$	40$	*True*
451234	Bail	85$	0$	*False*

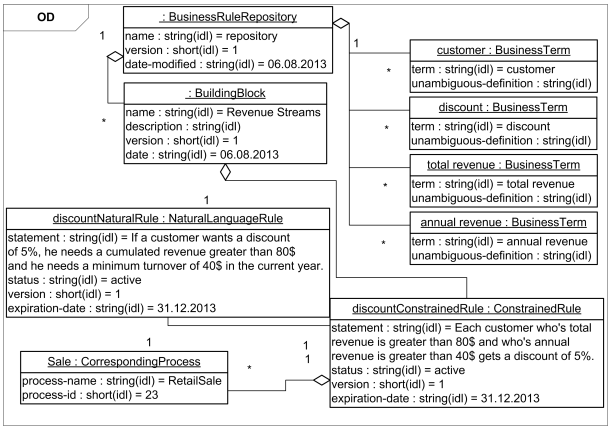

Figure 3: Ontology Object Diagram (excerpt)

2. **The necessary data is queried by the service via the D2RQ platform and passed to Jena.** The D2RQ engine maps a database into a RDF structure. Listing 2 represents the first database entry of table 2 as RDF transformed by the D2RQ engine.

Listing 2: Database in RDF

```
...
<rdf:Description
rdf:about=...#customer/124533>
<rdf:type rdf:resource=...customer/>
<rdfs:label>customer #124533</rdfs:label>
<j.0:customer_ID rdf:datatype=
"http://www.w3.org/2001/XMLSchema#integer">
124533
</j.0:customer_ID>
<j.0:customer_Name>Lewis
</j.0:customer_Name>
<j.0:customer_TotalRevenue rdf:datatype=
"http://www.w3.org/2001/XMLSchema#decimal">80
</j.0:customer_TotalRevenue>
<j.0:customer_AnnualRevenue rdf:datatype=
"http://www.w3.org/2001/XMLSchema#double">
40.0E0</j.0:customer_AnnualRevenue>
</rdf:Description>
...
```

3. **The rules are loaded by Jena.** Osterwalder et. al. divided a business model into 9 building blocks which can be represented by business rules. A rule within the building block "revenue stream" could be about determining a discount. This rule might look like the following: **R1:** Each customer who's total revenue *is greater* than 80$ and who's annual revenue *is greater* than 40$ *gets* a discount of 5%.

   Figure 3 illustrates an object diagram of the generic business ontology including rule **R1**.

[3] Total Revenue
[4] Annual Revenue

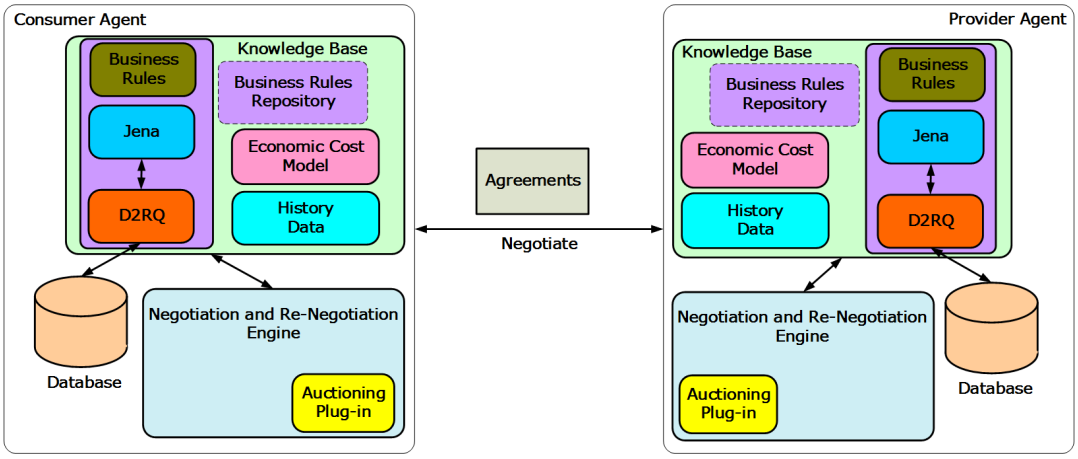

Figure 2: Negotiation Framework (simplified)

Rule **R1** has to be translated into a Jena Rule as shown in listing 3.

Listing 3: Discount Rule

```
@prefix j.0: ...
@prefix xs: http://www.w3.org/2001/XMLSchema#
[Discount:
(?customer rdf:type j.0:customer)
(?customer j.0:customer_TotalRevenue
?totalRev)
(?customer j.0:customer_AnnualRevenue
?annualRev)
ge(?totalRev,80)
ge(?annualRev,40)
->
(?customer j.0:discount '5'^^xs:integer)]
```

4. **The rules are inferred based on the RDF data from the D2RQ platform.**

   Jena is able to execute the rules represented in listing 3. The second data record of the database table represented in table 2 won't be affected as the annual revenue is lower than 40$. The first customer's revenues fulfill the limits. Therefore the costumer gets an additional element *discount* as shown in listing 4.

Listing 4: RDF Result

```
...
<j.0:discount rdf:datatype=
"http://www.w3.org/2001/XMLSchema#integer">5
</j.0:discount>
<rdfs:label>customer #124533</rdfs:label>
...
```

5. **The result of the inferring is RDF which can be queried via SPARQL to get the necessary data.**

   For querying results we can use SPARQL. Listing 5 shows such a SPARQL query gathering the discount of the the first costumer who has ID 33. This data can be used for further processing.

Listing 5: Query Discount

```
SELECT (str(?tmpdiscount) as ?discount)
 WHERE {?elments j.0:discount ?tmpdiscount.
?elments j.0:customer_ID ?id.
FILTER (?id =33)}
```

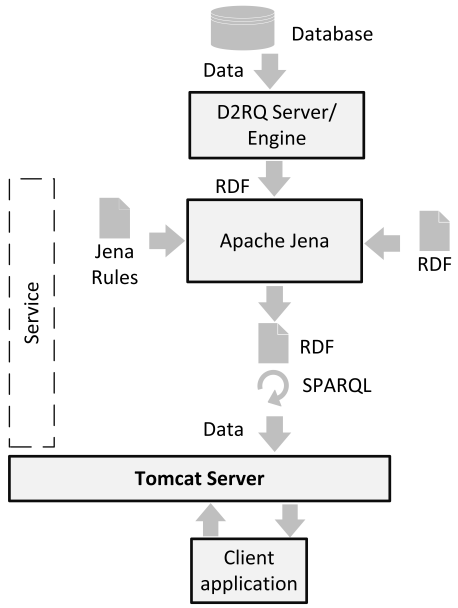

Figure 4: System Architecture

4.2 Blackboard Contracting Component

Contracting of specific services between service providers and service consumers is the basic necessity for constructing a concrete business workflow (service value chain), where each service of the abstract work flow has to be instantiated. This has to be done in a way that is optimizing (seeking for a minimum or a maximum) a custom **utility function**. In the focus of our economic domain this utility function will be

defined by the business strategy of the stakeholder and represents in turn a specific economic value/goal optimization. Mathematically, this can be mapped to a multi-dimension multi-choice knapsack problem. Several heuristics have been proposed to solve these QoS-aware service selection problems which are known as NP-hard.

Being aware of this computational complexity we decided to utilize a heuristic approach for optimize our business service value chains. A blackboard [6] – initially developed in the area of artificial intelligence – implements an A^*-algorithm to heuristically solve NP-hard problems. It is especially suited for complex problems with incomplete knowledge and uncertainties regarding the attributes and the behavior of the involved components.

A **global blackboard** represents a shared information space containing input data and partial solutions. The knowledge base representing the business strategy of an enterprise is composed of several independent regions, each resembles a single blackboard competence, both non-functional characteristics as costs, levels of reliability and security, etc., and functional characteristics, as APIs, protocols, etc. The global blackboard acts as a "mediator", allowing the different regions to communicate and work together finding the best solution (i.e. contract) according to the business strategy of the company.

The blackboard mechanism is listed in Algorithm 2. The goal function for evaluating possible service offer combinations for a request is called the *happiness function*. A decision tree is generated based on estimation of the happiness function for the visited paths. As shown in Algorithm 2, the expansion (choice) of promising service offers for a step in the request set, is handled by an `OpenList` and `BlockedList`. The `OpenList` contains a list of all possible service combinations to choose from. Each of these steps is rated by applying the happiness function that sums up the happiness of past decisions and the happiness of the next step. Considering this, the service which maximizes the happiness function is chosen for the next step in the optimization approach. The `BlockedList` contains services that do not fulfill the given requirements and therefore must be excluded from the set of possible solutions.

```
OpenList = expand(s1); BlockedList = [];
while OpenList ≠ [] do
    Act = best(OpenList); OpenList \= Act;
    if Act == Goal then
        return Act;
    end
    foreach d_x in expand(Act) do
        d_x.costs = Act.costs + h(d_x.costs);
        if d_x ∉ OpenList ∧ d_x ∉ BlockedList then
            < OpenList += d_x;
        else
            if d_x.costs < OpenList[d_x.id].costs then
                OpenList[d_x.id] = d_x;
            end
        end
    end
end
```

Algorithm 2: Blackboard Algorithm

We used this heuristic approach for several multi-dimension multi-choice knapsack problems in the area of computational science in the past [11, 21, 2] and it proved extremely feasible.

Delving into more details of the contracting process is beyond the scope of this paper. For more information see [20].

5. AN EXAMPLE WORKFLOW FOSTERING BUSINESS RULES

In this section we present exemplary a concrete implementation of the introduced workflow. First we introduce a running example for the following discussion: A grocery wants to introduce a loyalty card system. With every shopping a consumer can show his loyalty card to request a discount. This discount is calculated based on the total revenue and the annual revenue. Premium costumers get an extra, absolute, discount. The discount calculation is realized by business rules. The consumer data such as the total revenue is stored in an external database which is shown in table 2. In our example we used a MySQL database.

Figure 5 shows the solution of this problem using the architecture we introduced before.

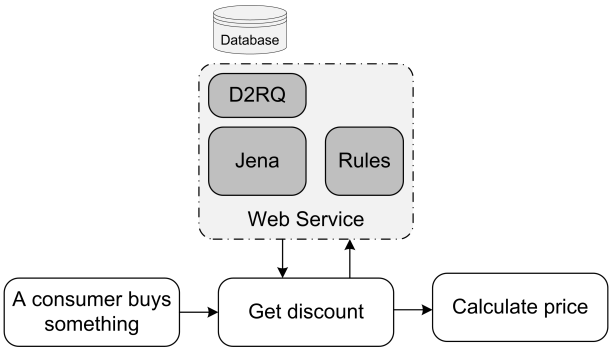

Figure 5: Buying Process

1. If a consumer buys something, the customer ID stored on the loyalty card is accessed by the loyalty card system. The loyalty card itself contains no further data. So if Lewis buys something, the loyalty card systems captures his customer ID 124533. Now the system has to calculate the discount based on Lewis's revenues as well as his premium status which is done by consulting a web service.

2. The customer ID is sent to a web service. We deployed the service on an Apache Tomcat server. Therefore we used the Apache Axis2 engine which supports the easy handling of web services.

   The listing 6 shows the Java web service call in the client application. First, a request has to be created. This is done be instantiating a new *GetDiscount* object. The class *GetDiscount* is defined within the web service. Before sending the request and consequently triggering the web service, the request variables must be set. In the listing, the customer ID, captured by the loyalty system, is passed to the web service as request variable.

Listing 6: Call a Web Service

```
// Creating the request
DiscountWebServiceStub.GetDiscount request =
    new DiscountWebServiceStub.GetDiscount();
request.setCustomerID(c.getID());

// Invoking the service
DiscountWebServiceStub.GetDiscountResponse
response = null;

try {
        response = stub.getDiscount(request);
} catch (RemoteException e) {
        e.printStackTrace();
}
```

The invoked *GetDiscount* web service has to connect to the customer database containing discount relevant data. In order to get the customer database in a RDF format, we use the D2RQ engine. Precondition for using the D2RQ engine is to create a mapping file. Listing 7 shows a code snippet of the mapping file coping with the database attribute TotalRevenue. The mapping file defines which attributes of the database should be transformed and consequently accessible as RDF. A tool for auto generating mapping files is provided by the D2RQ package.

Listing 7: D2RQ Mapping File

```
map:customer_TotalRevenue a
d2rq:PropertyBridge;
d2rq:belongsToClassMap map:customer;
d2rq:property vocab:customer_TotalRevenue;
d2rq:propertyDefinitionLabel
"customer␣TotalRevenue";
d2rq:column "customer.TotalRevenue";
d2rq:datatype xsd:decimal;
```

After the definition of the mapping file, the Java based web service is able to access the database via D2RQ as listing 8 shows. Therefore we use the class OntModel, provided by Jena, which represents an ontology model. In the listing, we create an empty ontology model inf. Afterwards we fill the model using the D2RQ engine. Based on the mapping file mapping.ttl, the D2RQ engine returns the MySql database table, shown in table 2, as RDF.

Listing 8: Accessing Database via D2RQ

```
OntModel inf =
ModelFactory.createOntologyModel(
"http://www.w3.org/2002/07/owl#");
inf.add(new ModelD2RQ("file:mapping.ttl"));
```

A simplified code snipped of the content of the ontology model inf is provided in listing 2.

As the database is mapped into RDF we are able to inference rules within the web service. Listing 3 shows an exemplary rule which is responsible for discount calculation. An additional rule for awarding an absolute premium discount for premium customers is shown in listing 9. This rule and the discount rule shown in listing 3 will be executed on the RDF data. Therefore the method *createInfModel* from the Jena API can be used.

Listing 9: Discount Rule

```
@prefix j.0: file:///D:/eclipse/vocab/
@prefix xs: http://www.w3.org/2001/XMLSchema#
[PremiumDiscount:
(?customer rdf:type j.0:customer)
(?customer j.0:customer_Premium ?premium)
equal(?premium,'true'^^xs:boolean)
 ->
(?customer j.0:PremiumDiscount
'2'^^xs:integer)
]
```

Jena returns the inferring result as RDF. So the customer shown in listing 2 gets an additional tag containing the discount and the absolute premium discount as listing 10 illustrates.

Listing 10: RDF Result

```
...
<j.0:PremiumDiscount
rdf:datatype=
"http://www.w3.org/2001/XMLSchema#integer">
2</j.0:PremiumDiscount>
<j.0:discount
rdf:datatype=
"http://www.w3.org/2001/XMLSchema#integer">
5</j.0:discount>
<rdfs:label>customer #124533</rdfs:label>
...
```

Now we have to extract the discounts of our customer from our RDF file and return it to the caller of the web service. Therefore we query the RDF result with the SPARQL statement shown in listing 11 and return the discount. Jena provides the class QueryFactory for executing SPARQL queries. Optionally, the web service caller can consume further web services using business rules.

Listing 11: Query Discount

```
SELECT (str(?tmpdiscount) as ?discount)
(str(?tmpPremiumDiscount) as ?premiumDiscount)
 WHERE {?elments j.0:discount ?tmpdiscount.
?elments j.0:premiumDiscount
?tmpPremiumDiscount.
?elments j.0:customer_ID ?id.
FILTER (?id =33)}
```

3. Finally the price can be calculated and charged by the grocery.

6. CONCLUSION

We introduced a business rules driven framework for consumer-provider contracting of web services. Using business rules increases maintainability and flexibility as the business user is able to manage the rules directly.

Currently, several standards, frameworks and technologies coping with business rules are available. As semantic web technologies are platform independent, widely accepted and open, we used these frameworks and technologies as basis for our architecture. Jena is one of these semantic web based frameworks and is able to process W3C standards. Additionally, Jena is able to infer Jena rules based on RDFs. The D2RQ platform allows to map databases such as MySQL and Oracle to RDF.

By building a prototype we demonstrated the functionality of our architecture based on Jena. To ensure flexibility, extensibility and scalability of the architecture, we implemented the prototype based upon the service oriented architecture using web services.

A full-fledged, ready-to-use implementation of our envisioned framework is ongoing. From research point of view we will analyse of the behavior of different auctioning models. Specific focus will be laid on defining business strategies in the knowledge base of the autonomic system. Thus, the framework aims for automatic, adaptive, and dynamic negotiation and re-negotiation processes establishing an ICT marketplace for services.

7. REFERENCES

[1] Alt, R., and Zimmermann, H.-D. Preface: Introduction to special section business models. *Electronic Markets 11*, 1 (2001), 3–9.

[2] Beran, P. P., Vinek, E., Schikuta, E., and Leitner, M. An adaptive heuristic approach to service selection problems in dynamic distributed systems. In *13th ACM/IEEE International Conference on Grid Computing (Grid 2012)* (Beijing, China, 2012), IEEE, pp. 66–75.

[3] Boyer, J., and Mili, H. *Agile Business Rule Development: : Process, Architecture, and Jrules Examples*. Springer Berlin Heidelberg, 2011.

[4] Browne, P. *JBoss Drools Business Rules*. From technologies to solutions. Packt Publishing, Limited, 2009.

[5] Chandrasekaran, B., Josephson, J. R., and Benjamins, V. R. What are ontologies, and why do we need them? *IEEE Intelligent Systems 14*, 1 (Jan. 1999), 20–26.

[6] Corkill, D. Blackboard Systems. *AI Expert 6*, 9 (January 1991).

[7] Drools. Drools - the business logic integration platform [online]. http://www.jboss.org/drools/. Accessed: 2013-06-23.

[8] Gordijn, J., and Akkermans, H. Ontology-based operators for e-business model de- and reconstruction. In *Proceedings of the 1st international conference on Knowledge capture* (New York, NY, USA, 2001), K-CAP '01, ACM, pp. 60–67.

[9] Graham, I. *Business Rules Management and Service Oriented Architecture: A Pattern Language*. Wiley, 2007.

[10] Hebeler, J., Fisher, M., Blace, R., Perez-Lopez, A., and Dean, M. *Semantic Web Programming*. Wiley, 2011.

[11] Kofler, K., Haq, I., and Schikuta, E. User-Centric, heuristic optimization of service composition in clouds. In *16th European Conference on Parallel and Distributed Computing (Euro-Par 2010)* (2010), P. D'Ambra, M. Guarracino, and D. Talia, Eds., vol. 6271 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, p. 405–417. 10.1007/978-3-642-15277-1_39.

[12] Mach, W., and Schikuta, E. A generic negotiation and re-negotiation framework for consumer-provider contracting of web services. In *Proceedings of the 14th International Conference on Information Integration and Web-based Applications & Services* (New York, NY, USA, 2012), IIWAS '12, ACM, pp. 348–351.

[13] Mach, W., and Schikuta, E. Toward an economic and energy-aware cloud cost model. *Concurrency and Computation: Practice and Experience* (2013), n/a–n/a.

[14] Maedche, A., and Staab, S. Ontology learning for the semantic web. *IEEE Intelligent Systems 16*, 2 (2001), 72–79.

[15] Mahal, A., and Zachman, J. *How Work Gets Done: Business Process Management, Basics and Beyond*. Technics Publications, LLC, 2010.

[16] Osterwalder, A., and Pigneur, Y. *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers*. Wiley Desktop Editions. Wiley, 2010.

[17] Pant, K., and Juric, M. *Business Process Driven SOA Using BPMN and BPEL: From Business Process Modeling to Orchestration and Service Oriented Architecture*. Packt Publishing, Limited, 2008.

[18] Ross, R. *Principles of the Business Rules Approach*. Addison-Wesley information technology series. ADDISON WESLEY Publishing Company Incorporated, 2003.

[19] Sharp, A., and McDermott, P. *Workflow modeling [electronic resource]: tools for process improvement and applications development*. Artech House, Incorporated, 2008.

[20] Vigne, R., Mach, W., and Schikuta, E. Towards a smart webservice marketplace. In *IEEE Conference on Business Informatics* (USA, 2013), IEEE.

[21] Vinek, E., Beran, P. P., and Schikuta, E. A dynamic multi-objective optimization framework for selecting distributed deployments in a heterogeneous environment. In *International Conference on Computational Science (ICCS 2011)* (Singapore, June 2011), Procedia Computer Science series, Elsevier Science.

[22] Von Halle, B., Barbara von Halle, L., Goldberg, L., and Zachman, J. *Business Rule Revolution (ebook): Running Business the Right Way*. Happy About, 2006.

[23] W3C. Owl web ontology language [online]. http://www.w3.org/TR/owl-features/, 2004. Accessed: 2013-06-23.

[24] W3C. Rdf primer [online]. http://www.w3.org/TR/rdf-primer, 2004. Accessed: 2013-06-23.

[25] W3C. Sparql query language for rdf [online]. http://www.w3.org/TR/rdf-sparql-query, 2008. Accessed: 2013-06-23.

[26] Witt, G. *Writing Effective Business Rules*. Morgan Kaufmann. Morgan Kaufmann, 2012.