Evaluation of Interoperability Issues between Business Rule Management Systems of IBM and SAP

Diploma Thesis

Submitted by

Ramin Zandipour

Matr.Nr.: 32249

TUHH Supervisors

Prof. Dr. rer. nat. Ralf Möller

Prof. Dr. rer. nat. Volker Turau

Company Supervisor

Khirallah Birkler

Declaration

I hereby declare that I have personally authored this report and that no other sources other than the ones listed at the end of the report have been used. The report is therefore, original and has not been published in Germany or in any other country. All the registered trademarks, names, and icons appearing in this report are the property of their respective owners.

Ramin Zandipour
31. January 2011

Acknowledgement

I am very grateful to Prof. Ralf Möller for helping me with the organization of the topic within a company and his guidance throughout the work. I also thank Prof. Volker Turau for accepting to review my work as a second supervisor.
My other special appreciation goes to the management of IBM Research & Development GmbH for providing me with the challenge to carry out my research work within the company and especially to Khirallah Birkler for his guidance throughout the work and his valuable feedback to my writings.

Table of Content

1 Introduction

The last few years companies have been facing a fundamental change regarding the structures of their IT-landscapes. To stand the competitive pressure in a globalized world it has been important for a company to become flexible and agile, which means to be responsive to change.
The responsiveness highly depends on the underlying IT-Systems, therefore the challenge is to align these systems with the business organization and it's goals. Visibility of how the business organization operates has to be created and functional silos have to be connected in order to support end-to-end business processes.

Whenever changes in the business strategy and business processes are decided the IT organization gets the instructions to implement these changes in the respective applications within the IT-Systems. This also includes changes in the policies and operational decisions associated with those policies, the so called "business rules". Because of the fact that all the code which represents the Business Rules is hard-coded in the applications, the IT developers have to do the tedious work of searching every single code related to the rules and modifying it. This results in a slow reaction to changes and there is also the risk that the code is not put in place properly, because the intention of the new policy has been misunderstood by the IT staff.

The possibility to source out the business rules and to manage them separately would solve these problems. It would increase the visibility of the applied policies and make them reusable across all applications and processes of a company – to name two benefits.
The environment where the policies can be sourced out is called a Business Rule Management System (BRMS).

With a BRMS the code for the policies is not only separated from the rest of the code, it also enables business users to bring in the decisions they make directly into the system with a user interface tailored to their skills.
This results in a noticeable increase of reactivity to changes and thus an important step is made towards an agile company whose competitiveness remains even in times of rapid change.

In the mid 2000's Business Rules Management Systems were offered by a variety of vendors, amongst others the french company ILOG, which was one of the market leaders of BRMS due to a well-engineered and fully developed management system. IBM acquired ILOG in 2006 to complement their middleware-products with a BRMS, so that they could offer a complete software solution for companies to become flexible and agile.
The leading vendor of ERP-Software, SAP, offers two BRMS. One is based on their proprietary programming language ABAP and one is based on Java . The Java based BRMS has it's origin in the acquisition of the Indian BRMS vendor Yasu in 2007, whose BRMS, called “QuickRules”, was then integrated into the SAP portfolio. The ABAP based BRMS was developed by SAP itself and it's first version was released in 2008.

The objective of this thesis consists of three tasks:

1. Exploring the general functionality of the BRMS offered by IBM and SAP
2. Describing how the BRMS of IBM and SAP are embedded in their respective ecosystem
3. Evaluating the the interoperability aspects between these systems, with focus on the integration of IBM's BRMS with SAP's system (BRMS and ecosystem). Also elaborating to which extent such an interoperability is feasible.

2 IBM and SAP relationship

This chapter starts with an introduction of the companies IBM and SAP, whose middleware products are the subject of this thesis. The general relationship between IBM and SAP is discussed, with focus on the question in which degree they have a partnership or competition and how the history of this relationship looks like.

Finally an already existing interoperability between middleware software of IBM and SAP is shown through their Enterprise Portals.

2.1 Company Profiles

Wikipedia gives the following definitions of the companies:

"**International Business Machines** (**IBM**) is a United States multinational technology and consulting firm headquartered in Armonk, New York. Founded in 1911, IBM manufactures and sells computer hardware and software, and it offers infrastructure, hosting and consulting services in areas ranging from mainframe computers to nanotechnology. "[1]

"**SAP AG** is a German software corporation that provides enterprise software applications and support to businesses of all sizes globally. Headquartered in Walldorf, Germany, with regional offices around the world, SAP is the largest enterprise software company in the world (as of 2009). It is also the largest software company in Europe and the fourth largest globally. The company's best known products are its SAP Enterprise Resource Planning (SAP ERP) and SAP BusinessObjects software. "[2]

The following table gives an overview of both companies:

	IBM	**SAP**
Industry	Computer Systems, hardware, software, consulting and IT Services	Computer Software
Founded	Endicott, New York June 16, 1911	Weinheim, Germany (1972)
Headquarters	Armonk, New York	Walldorf, Germany
Employees	399,409 (2009)	47,578 (2009)
Revenue	99,9 billion (2010)	10,671 billion (2009)

Table 2.1.: Overview of IBM and SAP [1][2]

2.2 IBM and SAP relationship

Generally there are three layers of relationship between IBM and SAP, which will be explained in this chapter:

They are

- Partners
- Competitors
- Customers

Partners:

Today IBM and SAP share more than 9000 customers. This is because in the 90's and early 2000's IBM and SAP offered products to companies which were completely complementary, since there was no overlap between their functionality. IBM had it's focus on delivering hardware and infrastructure software, while SAP focused on delivering ERP business applications to the customers. So if customers wanted to use an ERP handling their business, which resides on a stable and reliable IT Infrastructure, they often chose the combination of IBM's infrastructure and SAP's ERP, which consequently led to a strategic alliance between both companies.

This partnership also led to a common support and education service for the building up of skill on both fields, which provided another reason for companies to adopt this solution.

Competitors:

With the introduction of the Service Oriented Architecture (see next Chapter for details) in the early 2000's SAP realized that there was a need to connect the different processes of it's ERP modules, including for example Customer Relationship Management (CRM), HR (Human Resources) or SCM (Supply Chain Management), internally to enable automated "straight-through"-processing across the modules. Also the seamless integration of third party applications was demanded by customers. Therefore SAP decided to create it's own middleware components to offer the necessary infrastructure for companies to connect applications, which were built on different platforms.

Because the functionality of connecting disparate systems had also been already covered by IBM's middleware products, competition in the infrastructure area was created between IBM and SAP, represented by the products of the IBM's WebSphere brand and SAP's NetWeaver brand.

Nevertheless the most important factor in what to deliver to the customer is to determine what the customer itself needs and wants. Therefore another **partnership** of IBM and SAP in the infrastructure was established because it turned out that their infrastructure products had different strengths and were complementary.

SAP's NetWeaver middleware products were perfectly capable of integrating the ERP modules from SAP's Business Suite so that they could inter-operate in a straight-through process, while this could be handled by IBM's middleware only through special adapters. So if a customer just wanted to integrate it's already existing ERP modules without the integration of other applications the choice was to buy SAP's NetWeaver integration products.

However, IBM was already very experienced in creating and delivering middleware products covering all kinds of integration demands of customers. It was also verified through tests, that in terms of reliability, security and scalability the WebSphere products were better than the newly established NetWeaver products [3]. As a result, if the customer puts priorities on reliability, security and scalability in the integration of disparate systems, their choice would then be to buy IBM's middleware products.

The interoperability of IBM and SAP in the infrastructure area was also needed for customers who wanted both. They already had their ERP applications running with SAP products and also wanted to integrate third party applications to enable automated enterprise-wide business processes – even beyond the boundaries of the enterprise. Although NetWeaver was also able to integrate third party applications the mentioned performance aspects led to the decision of choosing WebSphere. As a consequence, the ERP modules were integrated by NetWeaver which itself was integrated in the WebSphere infrastructure. The following picture gives an example of this scenario, where 3 integration scenarios are illustrated:

1. SAP's Software for application integration, SAP NetWeaver Exchange Infrastructure (SAP XI), connects SAP's modules. Simultaneously SAP XI itself is connected to IBM's WebSphere middleware through open standards.
2. Each module of SAP is connected separately to WebSphere through an IBM SAP adapter, which handles the communication aspects.
3. Connection of a legacy application to SAP XI through an adapter provided by SAP.

All applications, including IBM WebSphere and SAP XI, can be monitored using IBM's "Tivoli" software.

Please note, that SAP XI has been enhanced and renamed to SAP PI, which is short for "SAP Process Integration".

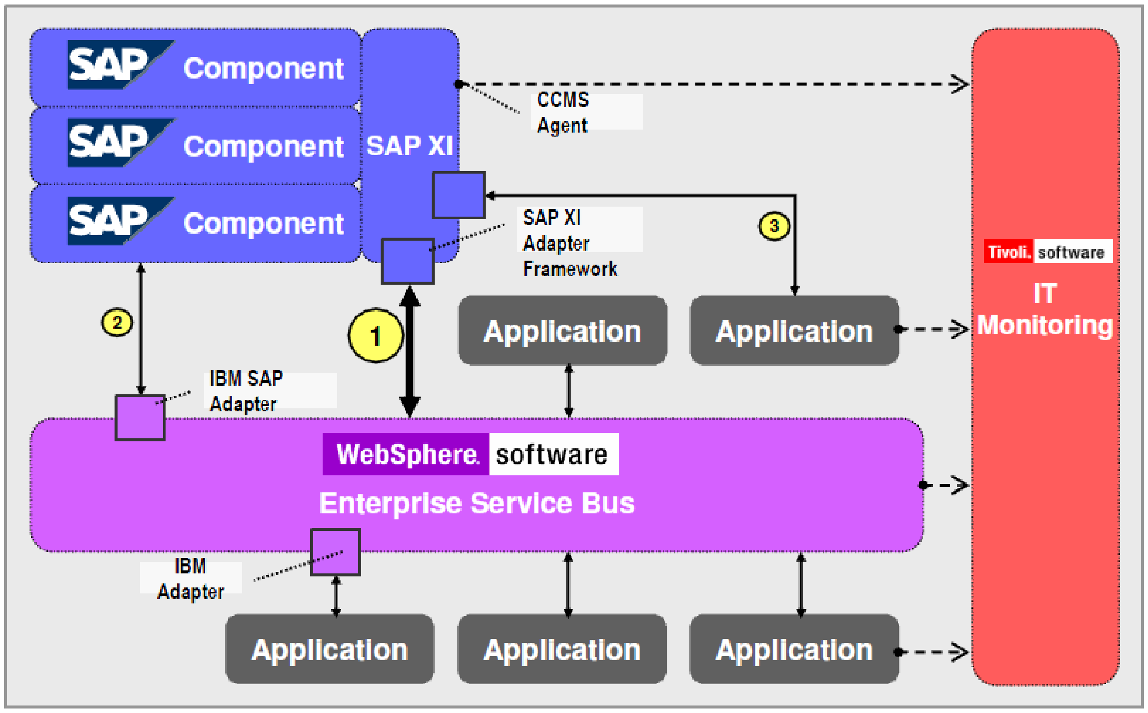


Fig. 2.1: Interoperability of IBM and SAP in the field of application integration [3]

Customers:

Since both IBM and SAP are market leaders in their core business, they are also customers of each other. While SAP has invested in IBM's hardware and middleware products, IBM is creating unified company-wide business processes using SAP's business applications. The IBM internal SAP implementation ranges among the TopTen installations of SAP. [3]

This was an overview of the relationship between IBM and SAP. It could be seen, that the interoperability of the BRMS of IBM and SAP which is the subject of this thesis is part of a strategic alliance between both companies. The next chapter will exemplify how an existing interoperability between middleware components of IBM and SAP is established technically.

2.2 Interoperability between Enterprise Portals

An enterprise portal offers customers a user interface where the services of the enterprise applications of a company can be used. A portal is made up of portlets, where each portlet defines a service or information offered by an application in the enterprise – or from other sources, like the internet.

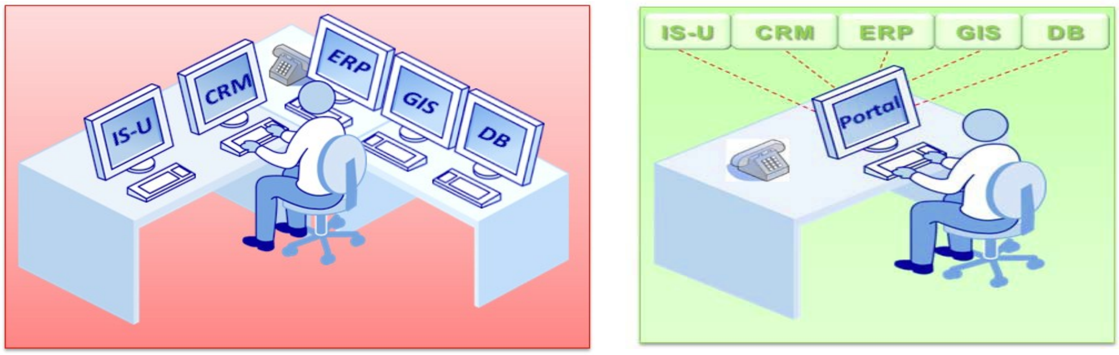
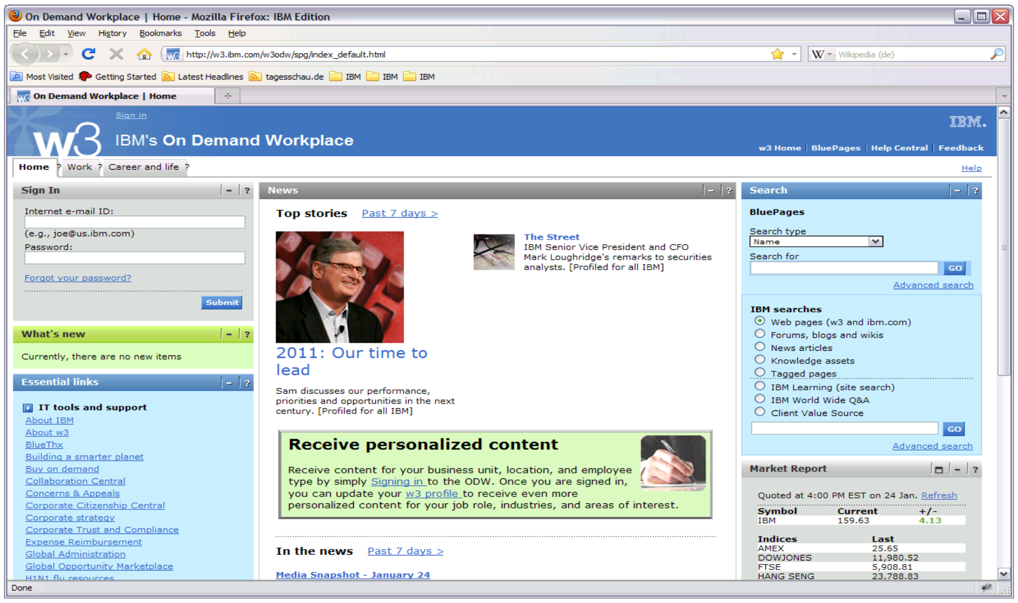


Fig 2.2: SAP's Enterprise Portal enables a unified User Interface for it's modules [4]

Here is an example of a portal interface, which is offered by IBM to it's employees:


On Demand Workplace | Home - Mozilla Firefox: IBM Edition
File Edit View History Bookmarks Tools Help
Wikipedia (de)
Most Visited
Getting Started
Latest Headlines
tagesschau.de
IBM
IBM
IBM
On Demand Workplace | Home
Sign in
w3
IBM's On Demand Workplace
IBM.
w3 Home | BluePages | Help Central | Feedback
Home ? Work ? Career and life ?
Help
Submit
GO
GO
Done

http://w3.ibm.com/w3odw/spg/index_default.html

Sign In

Internet e-mail ID:

(e.g., joe@us.ibm.com)

Password:

Forgot your password?

What's new

Currently, there are no new items

Essential links

IT tools and support

About IBM
About w3
BlueThx
Building a smarter planet
Buy on demand
Collaboration Central
Concerns & Appeals
Corporate Citizenship Central
Corporate strategy
Corporate Trust and Compliance
Expense Reimbursement
Global Administration
Global Opportunity Marketplace
H1N1 flu resources

News

Top stories Past 7 days >

2011: Our time to lead

Sam discusses our performance, priorities and opportunities in the next century. [Profiled for all IBM]

The Street
IBM Senior Vice President and CFO Mark Loughridge's remarks to securities analysts. [Profiled for all IBM]

Receive personalized content

Receive content for your business unit, location, and employee type by simply Signing in to the ODW. Once you are signed in, you can update your w3 profile to receive even more personalized content for your job role, industries, and areas of interest.

In the news Past 7 days >

Media Snapshot - January 24

Search

BluePages

Search type: Name

Search for

Advanced search

IBM searches

[✓ Web pages (w3 and ibm.com)]
Forums, blogs and wikis
News articles
Knowledge assets
Tagged pages
IBM Learning (site search)
IBM World Wide Q&A
Client Value Source

Advanced search

Market Report

Quoted at 4:00 PM EST on 24 Jan. Refresh

Symbol	Current	+/-
IBM	159.63	4.13

Indices	Last
AMEX	25.65
DOWJONES	11,980.52
FTSE	5,908.81
HANG SENG	23,788.83

Fig 2.3: IBM's enterprise portal for it's employees

Reasons:

Decisions in a company for a specific software in the past have often been made on a departmental level rather than an enterprise level. The advantage of this approach was that every department was using software which was tailored to their specific needs and requirements. The main problem occurred, when departments were consolidated or the company decided to use one kind of software for all departments.

So when departments were put together, the enterprise portals also had to be put together. Deciding for one portal software and refusing the other one was not a good idea, since the portals had overlapping functionality, but were not identical. The IBM portal was more suitable for one department, while SAP's portal was suitable for the other department. Also the development environment for the portlets was different, so money had to be invested in building up the necessary skills.

Accomplishment:

IBM and SAP decided to enable the interoperability between their portal products, so that customers would be able to use both instead of making an either-or decision. They would for example be able to choose IBM's portal solution as access point for external users and SAP's portal solution as access point for internal employees.

Two main interoperability scenarios were performed: The "portal-hub-scenario" and the "portal-in-portal approach" [5].

In the portal-hub-scenario each portal in a company was an application-specific island and the goal was to export portlets from one portal and import them in another, so that they could interact.

The portal-in-portal approach was necessary, when the company for example decided to use one portal as the user interface, while the other portal then provided the functionality needed to display the content on the first portal.

Besides, industry-driven standards made it possible to technically replace one portal solution with another through standard access in the development of the portlets. So independent of the fact whether an IBM or SAP portal solution was deployed, both IBM and SAP implemented the same API for accessing their portlets. Two industry-wide standards making this possible are JSR-168 and WSRP (Web Services for Remote Portlets).

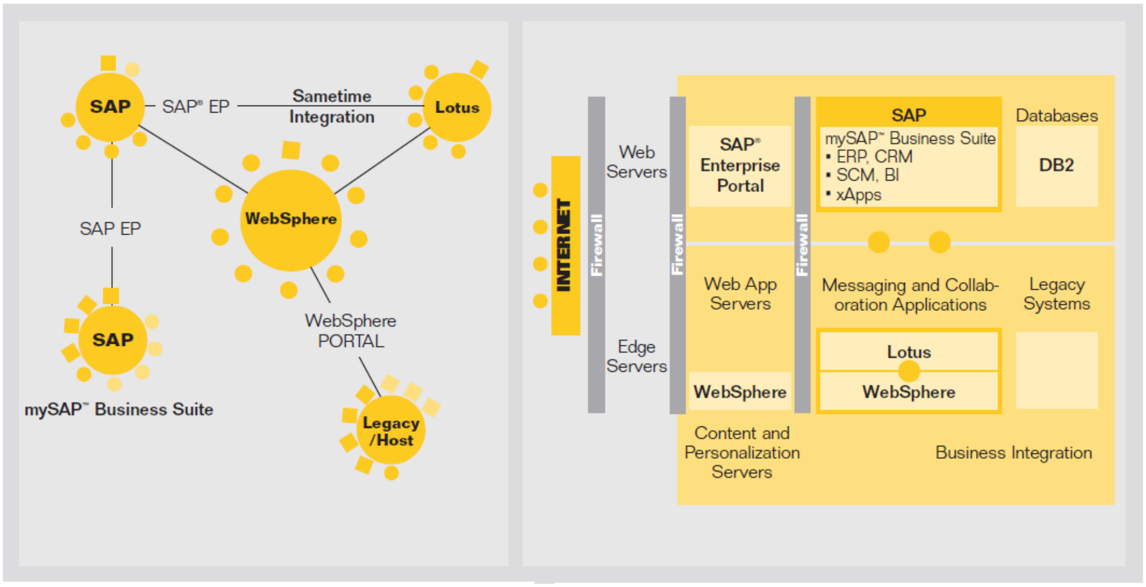


Fig 2.4: Illustration of Portal-Hub-Scenario on the left side and Portal-in-Portal approach on the right side [5]

3 Service Oriented Architecture

(The following chapter partially orientates on [6])

As mentioned in the introduction of this thesis, companies are facing the challenge to be able to change business processes quickly whenever it is needed. This is only possible when the underlying IT-systems are aligned well with the business processes which are modeled by people from the business staff. This alignment was not available in the 90's, which resulted in a slow adoption of changes in a business process. The model of the business process had to be translated into a model suitable for the IT staff, which could then program or modify the code in the respective applications. The result was an imperfect adoption of the changes, due to losses in the translations and misunderstandings between the business and IT staff.

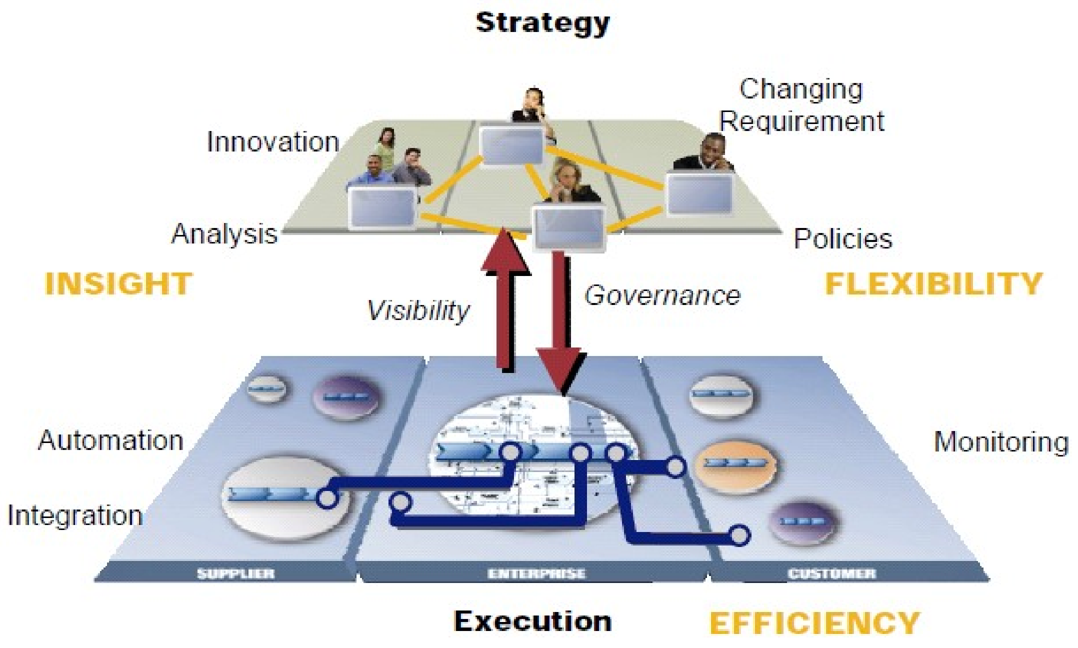


Fig. 3.1: Aligning Business with IT [7]

Nowadays there are tools for supporting the so called “Business Process Management” (BPM). BPM has the vision, that every change made in a business process is automatically applied by the underlying IT-Systems with minimum involvement of the IT staff. This vision has not been reached yet, but companies have made a huge step in that direction by applying the principles of the “Service Oriented Architecture” (SOA) to their IT-Systems.

This chapter explains what exactly the principles of a Service Oriented Architecture are and how this architecture historically evolved.

3.1 Point-to-Point Communication

In the 90's the decisions of which IT software was going to be used in a company were mostly made on a departmental level rather than on enterprise level. The result was a heterogeneous IT environment, where each department used different applications, built on different platforms, to solve it's problems.

The approach to integrate two applications was first made by creating a point-to-point connection through the implementation of the necessary interfaces. This was relatively easy to realize with the involvement of low costs. But with an increasing number of point-to-point integrations the shear complexity of the resulting integration landscape hindered the IT's ability to to respond to change and additionally implied the following drawbacks :

- There was a limited management infrastructure to provide management capabilities of the integrations, like monitoring or change control.
- Without the management capabilities, it was not visible of what was actually deployed. Duplicated interfaces or interfaces, which were deployed but not used anymore, existed.
- The modification or enhancement of existing interfaces was difficult
- The interfaces were proprietary, standards did not exist
- As the development staff moved from project to project the necessary skill to support the existing deployment became transient.

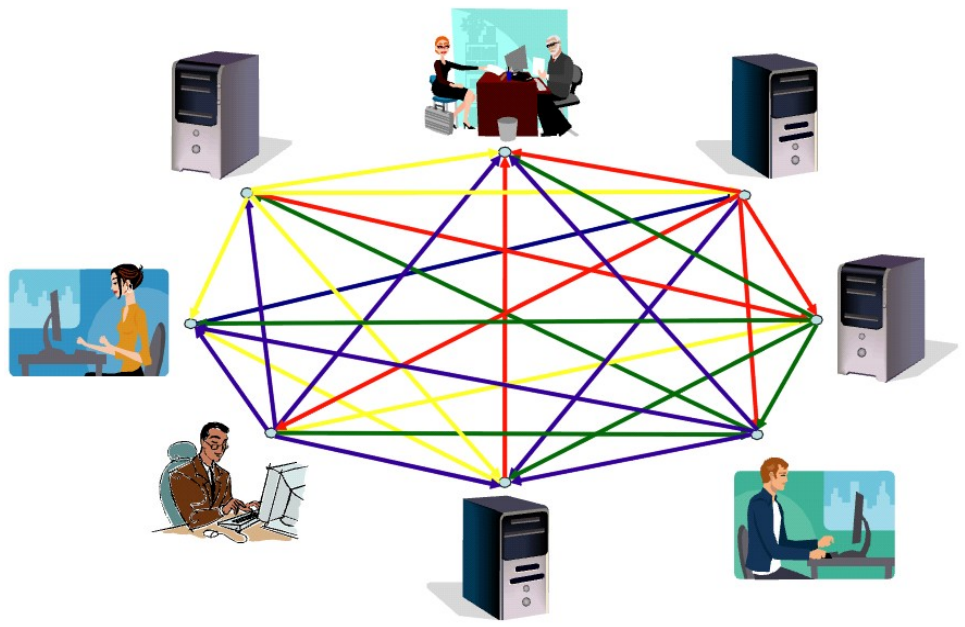


Fig. 3.2.: Enterprise applications connected through point-to-point integrations[6]

Summarized, there was a very low level of control over the existing integration activities, because there was a lack of governance capabilities through an integration framework.

3.2 Enterprise Application Integration

The next step in the evolution of application integration was the development of a framework, which enabled the management and governance of existing integrations using standards and was known as "Enterprise Application Integration" (EAI).

In an EAI scenario, the applications of an enterprise communicate through a centralized message broker, which was responsible for the mediation of messages between the applications. The backbone for EAI was "Message Oriented Middleware" (MOM), which allowed physically decoupled applications to exchange data in real time and initiate functions through a reliable messaging infrastructure. The applications were connected to the message broker through adapters, which converted the messages from the message format of the calling application to the message format of the target application. This also led to the appearance of a complete industry providing adapters for applications, which could consume or expose data in the form of messages. This integration pattern of EAI with a centralized message hub is called “Hub-and-Spoke” model.

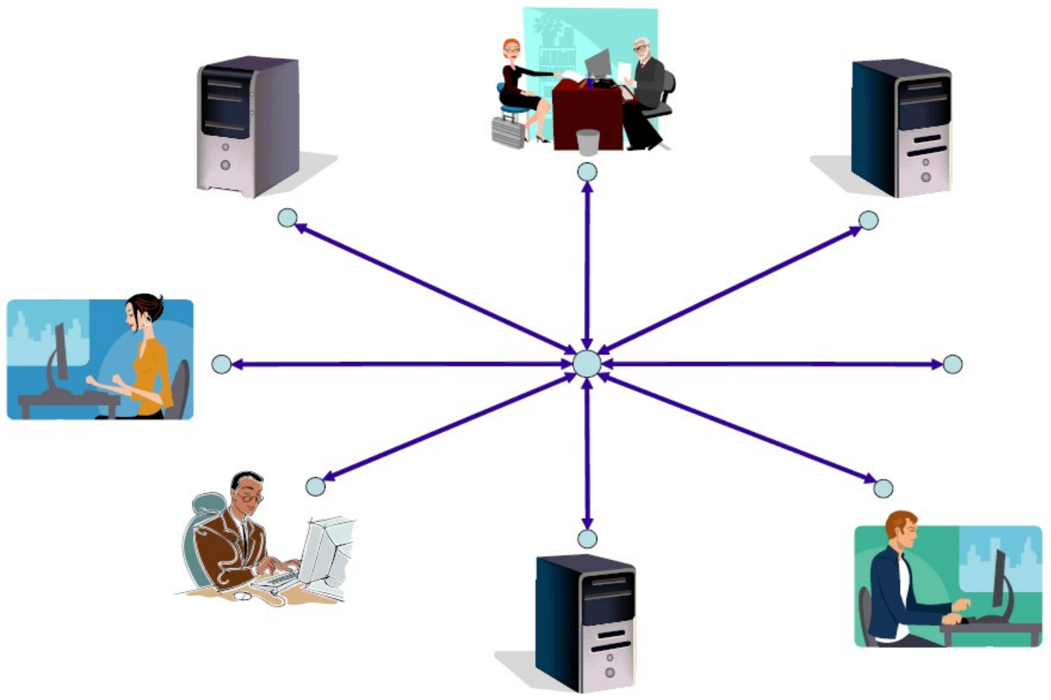


Fig. 3.3: Integration of enterprise applications through a Hub-and-Spoke model[6]

While in a point-to-point integration approach the number of necessary interfaces could reach n*(n-1)/2, the EAI approach needed n interfaces. For the integration of 50 applications, this means a difference between 1225 and 50 interfaces.

Some other key advantages provided by the implementation of EAI are:

- Business changes like acquisitions or the merging of departments could be adapted, due to a fast unification of the applications involved
- User applications could be presented through Enterprise Portals, which provide a unified view of the information, therefore enabling better decision making
- Data consistency across all applications in a company, which reduces latency

Despite the great improvement of application integration through the Hub-and-Spoke model it had two major disadvantages:

- The scalability of the mediation hub was limited since it depends on the hardware.
- Managers hesitated to invest in EAI, because EAI solutions were about capability enablement, therefore it was difficult to provide an ROI justification.

The second reason was basically due to the fact, that EAI was an IT driven initiative and without a clear executive and business buy in the realization was hardly possible.

Nevertheless, by the late 90's and early 2000's there was a shift in the thinking of how to apply EAI. The focus was now put on the alignment of the integration with the company's business processes. It was being exploited how the EAI capabilities enabled the building of "straight-through" processes, that linked the business processes which were embodied in disparate applications.

This was the beginning of decoupling the applications through the concept of service, in which the source end-points were defined as service consumers and the target end-points as service providers. Service providers offered a certain functionality, which they could perform, and service consumers could make use of this functionality. Both service consumer and service provider had to map their requirements to abstract service specifications which enabled the hiding of implementation details.

An IT architecture, where the functionality of the applications is exposed as services, is called a "Service Oriented Architecture".

An SOA generally comprises the following elements:

- **"Service**: An abstract specification of an interface (function and data), that hides the implementation details of the service provider from the service consumer.
- **Policy**: The constraints and capabilities that apply to a service, including security and Quality of Service (QOS).
- **Granularity**: The specification of services that are of the appropriate granularity for the architectural context-
- **Governance**: An organizational structure that can define and implement roles, responsibilities, processes, policies, practices, principles, guidelines and frameworks for supporting service consumption and provisioning.
- **Standards**: Adoption of standards that underpin the interoperability of services within the architectural context.

– **Responsiveness**: Focused on the creation of service specifications in a just-in-time manner.

– **Discovery**: A mechanism that enables the publication and discovery of service definitions.

– **Composition**: The ability to construct software solutions through the composition of services.

– **Platform**: A technical infrastructure that provides the fabric for connecting a service consumer with service provider(s)."

A technical platform, with which an SOA can be realized, is the so called "Enterprise Service Bus":

> "In computing, an **enterprise service bus** (ESB) is a software architecture construct which provides fundamental services for complex architectures via an event-driven and standards-based messaging engine (the bus). Developers typically implement an ESB using technologies found in a category of infrastructure products, usually based on recognized standards." [Source:Wikipedia]

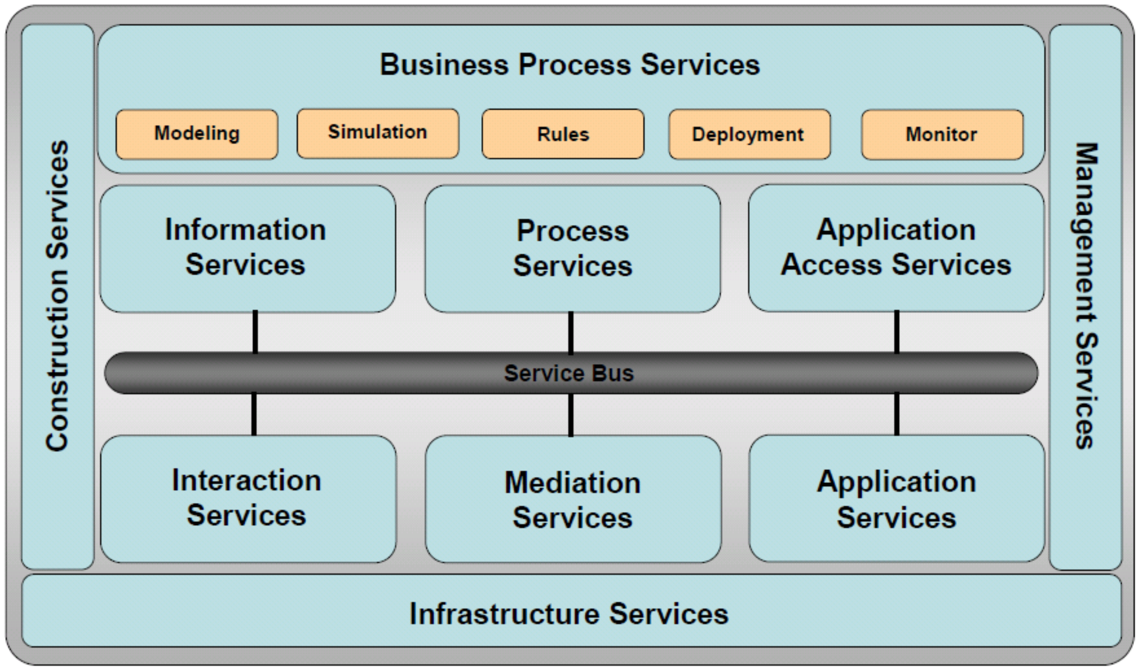


Fig. 3.4.: Reference model of the implementation of an SOA through a Service Bus[6]

The fundamental services mentioned in the definition are the following:

– **“Information Services**: The ability to expose data through the federation, replication and transformation of data sources.
– **Process Services**: The ability to create services that are based on the orchestration of other services into a stateful or stateless process flow.
– **Application Access Services**: The ability to expose application function through adapters and connectors.
– **Application Services**: The ability to expose component interfaces within contemporary application server platforms.
– **Process Services**: The ability to create services that are based on the orchestration of other services into a stateful or stateless process flow.
– **Interaction Services**: The ability to interface with interaction technologies, including UI and business partners.
– **Mediation Services**: The ability to perform mediation actions such as transformation and routing.
– **Management Services**: The ability to manage, service infrastructure including deployment, monitoring, security, change control, availability etc.
– **Infrastructure Services**: Provision of the basic IT infrastructure to support the service implementations, including plumbing and protocols.
– **Construction Services**: Provision of tools and technologies to develop/configure, test and validate service elements”

An SOA built on an ESB finally provides the flexibility which is necessary for the successful alignment of Business and IT, encompassing process, infrastructure and people.

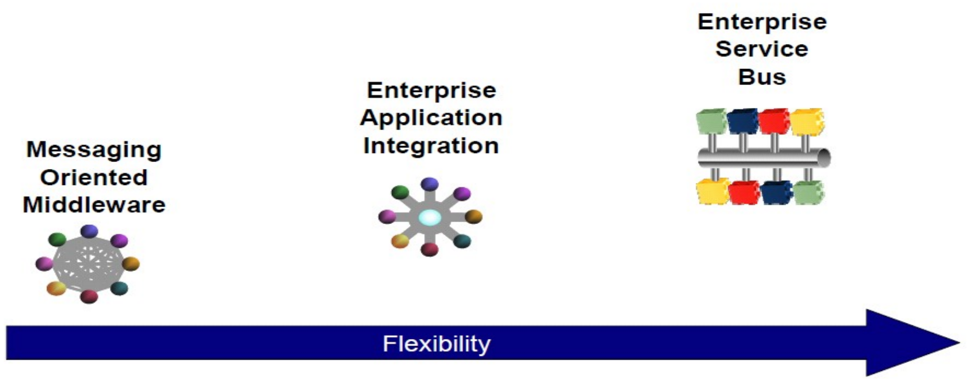

Messaging Oriented Middleware
Enterprise Application Integration
Enterprise Service Bus
Flexibility

Fig. 3.5: The evolution of platforms supporting integration architectures[6]

Roles in an SOA:

The key components in an SOA are the messages that are exchanged, end-points that consume or provide services and shared transport mechanisms that allow the flow of messages. Therefore in an SOA three critical roles exist:

- **Service Provider:** Allows access to services, creates a description of a service and publishes it to the service broker.
- **Service Consumer:** Is responsible for discovering a service by searching through the service descriptions given by the service broker. A requester is also responsible for binding to services provided by the service provider.
- **Service Broker:** Hosts a registry of service descriptions. It is responsible for linking a service consumer to a service provider.

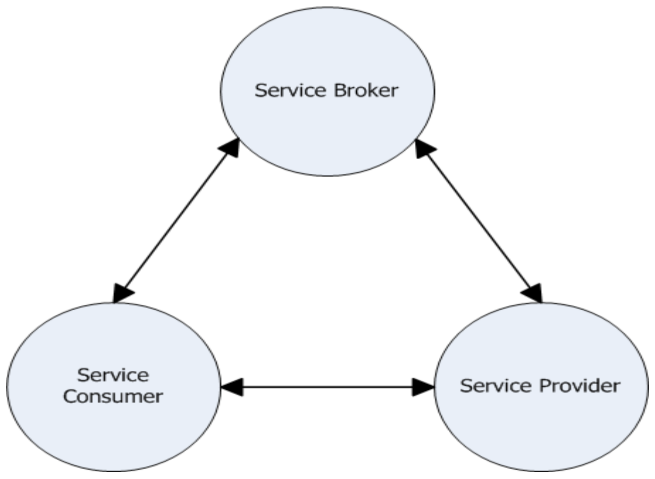


Fig. 3.6: The SOA triangle

3.3 Web Services and related Standards

The adoption of an SOA is not tied to a particular technology or product set. But to reduce the amount of proprietary elements and to enable seamless interoperability between the disparate applications the implementation of industry standards is key. Today's most relevant implementation of a Service in an SOA is the Web Service with it's related standards.

A definition of Web Services is the follows:

> "A Web service is a software application identified by a URI, whose interface and bindings are capable of being defined, described and discovered as XML artifacts. A Web service supports direct interactions with other software agents using XML-based Messages exchanged via internet-based protocols." [Source:W3.org]

Numerous vendors have contributed to the development of the standards, which relate to web services, including IBM, SAP, Tibco, Sun and BEA. The conduction of these standards is performed by three organizations:

- **"W3C**: The World Wide Web Consortium has been responsible for the promotion and development of standards that ensure the interoperability of the Web. W3C has responsibility for the core standards of Web Services including the core XML standards.
- **OASIS**: The Organization for the Advancement of Structured Information Standards has a strong focus on e-business standards and has a major role in the development of supporting Web Service standards.

- **WS-I**: The development of standards for Web Services takes time and is in a constant state of evolution, with the additional complication that vendors will extend their implementation of a standard to accommodate specific enhancements within product that is yet to be adopted (if at all) by the relevant standards body. WS-I in effect provides a service to the consumers of standards by publishing a baseline of standards interoperability at a given point in time.

Some key standards, which relate to Web Services, are the following:

- **XML**: Extensible Mark-up Language is a tagged based language for describing data and documents and is the foundation syntactic form for all things Web Service
- **WSDL**: Web Services Description Language. XML based language for describing a Web Service including the location, operations, inputs & outputs and the access mechanism.
- **SOAP**: An XML based wire protocol for the exchange of structure and content between peers in a distributed environment. It is the base wire protocol for describing and exchanging Web Service interactions between consumers and providers.
- **HTTP**: Hypertext transfer Protocol is the base communications protocol that is used to exchange text and media in a platform and application independent manner. HTTP has become the backbone protocol for the Internet and was the first communications protocol supported for Web Services.
- **UDDI**: Universal Description, Discovery and Integration is a directory model that specifies how Web Services and other artifacts are described, published and discovered.
- **WS-Security**: Web Services Security is an extension to the SOAP specification for describing the security constraints and capabilities such as QoS for Web Services.
- **XML Schema**: XML Schema Definition language is an XML based syntax for describing the structure and content of an XML document that can be used for example to validate the contents of an XML message.

– **WS-I**: Web Services Interoperability Basic Profile that provides a set of guidelines for the use of selected Web Service standards that will achieve interoperability across product vendors and platforms. "

SOA triangle with Web Services:

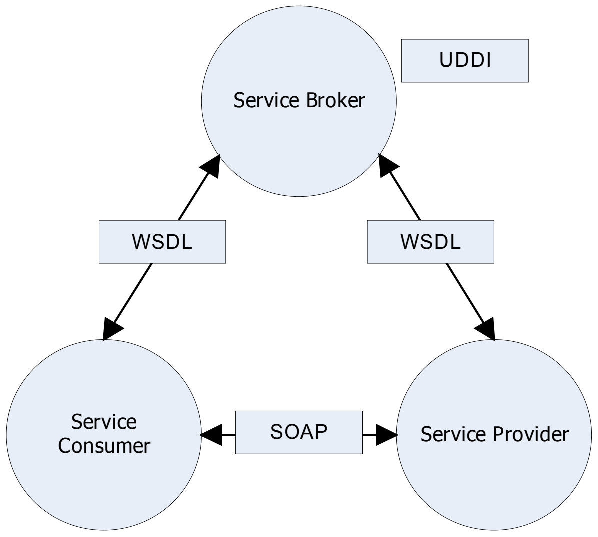


Fig.3.7.: The SOA triangle with Web Services

A common scenario for consuming a Web Service is the following:

1. The Service Provider generates a WSDL file, where it is described, through which address the service can be accessed and what the input format and the format of the expected output data looks like.
2. The WSDL file is then published at the UDDI so that it can be looked up by anyone having access to the UDDI. The necessary information for calling the service is then saved in the UDDI as meta-data.
3. The Service Requester checks the UDDI for the desired service and receives the WSDL file.
4. The Service Requester puts the required input data in an XML file in the SOAP format and sends it to the address of the Service Provider.
5. The Service Provider receives the request, computes the result and puts it in a SOAP file. He then returns the SOAP file to the Service Requester.

6. The Service Requester, who knows how the format of the result-data looks like, receives the SOAP file, unpacks it and uses it for his process or application.

Web Services Protocol Stack:

The Web Services Protocol Stack describes the layering of the set of internet protocols used to design, implement and discover web services. While the most commonly used protocol for the transport layer is HTTP, the other layers are all based on the XML-standard, which has been described above together with the other standards.

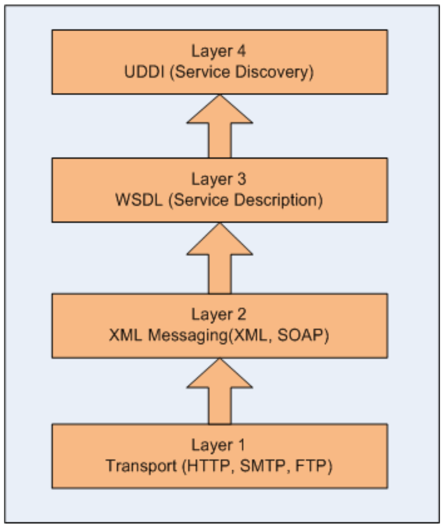


Fig. 3.8: Layers of the Web Services Protocol Stack

This chapter provided a brief historical and technical overview of the evolution in application integration concepts from the beginning until today. The SOA paradigm proved to be more than a hype, nevertheless new concepts for IT architecture can be introduced in the future, which will serve market demands better than the SOA approach, but most probably there will be an evolution rather than a revolution.

4 Business Rule Management Systems

As mentioned in the introduction of this thesis Business Rules are representations of the policies and regulations which exist in a company.

The Business Rule Group gives the following definition of a Business Rule:

> *"A business rule is a statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behavior of the business. The business rules that concern the project are atomic -- that is, they cannot be broken down further. "*

They are usually prone to change and the speed with which changing policies are adopted in the IT-Systems of an organizations is a crucial factor for the company's agility, and therefore competitiveness and success.

The issues which are responsible for a slow adoption of policy changes in companies where a BRMS is not leveraged are manifold. Every business rule of an organization is hard-coded in the business applications and are therefore hidden and isolated, which makes the maintenance and change a tedious work for IT developers which again costs time and money.

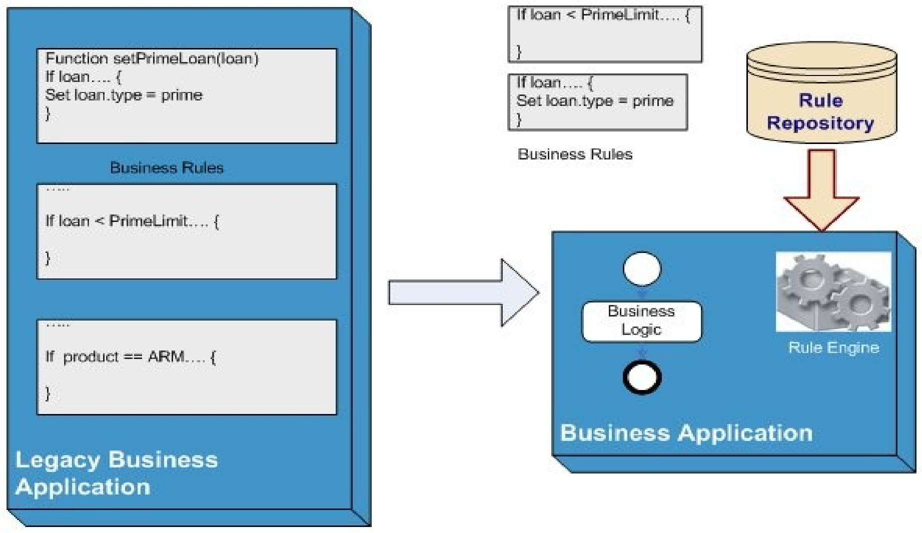


Fig. 4.1: Separating the business rules from applications enables their separate management [8]

It is also common that in multiple applications of a company the same rule is applied, which means that changing a rule results in changing the same rule in every application where this rule has been programmed in. Besides the reduced reactivity to changes, the problem of this procedure is, that the more applications use the same rule, the more probable it is that some rules have not been changed correctly. The result is that multiple versions of the same rule exist over time. Sourcing out the rules from the applications using them and managing them separately makes them reusable, which eliminates the risk of having multiple versions of the same rule.

Another important issue is that changes which are decided by the business staff cannot be easily tested or simulated. This is due to the fact that changes have to be implemented at every place the code resides and then the whole application has to be recompiled and tested. And the fact that rules change much more frequently than applications make it a very ineffective procedure to recompile the application whenever changes in the rules occur. This makes it risky to conduct policy changes, and also the business staff is not given the opportunity to easily find out which changes in the policy lead to which effects in "what if"-scenarios.

So all in all the situation in companies which do not make use of a BRMS is that they lack organizational agility and the productivity of their IT staff is not as high as it could be if the rules were maintained directly by the responsible managers.

A Business Rule Management System solves these problems and the concepts are described in the following.

A Business Rule Management System is made up of at least the following components:

- An environment where rules can be defined, analyzed and maintained
- A repository, where rules are stored and shared
- A runtime environment, where the defined rules can be executed using a rule engine

A BRMS sources the rules which reside hard-coded in the application out to a single environment where the rules can be defined, analyzed and maintained. The tacit knowledge of the people who own the rules, the applications, processes and documents which contain the rules are all stored and shared using a central repository.

In its primitive form, a BRMS only offers an environment for defining and managing rules for technical users, which means that the rules are defined using a language which is not understandable to business users.

Advanced BRMS, however, also offer a graphical user interface tailored to the skills of business users where the rules can be authored and managed by their true owners. They can even be managed using familiar tools for business users, like Microsoft Word or Excel.

Providing rule management capabilities to the business staff offers great advantages with regards to the responsiveness to changes:

- The reliability on IT departments is greatly reduced, which also reduces the risk, that rules are not understood correctly by the IT staff and therefore implemented incorrectly.
- Rules are much faster adopted into the production system, because the rules are defined and changed directly by the people who decided them.
- Business users can test the effects of new policies directly in a simulation, because their environment is coupled with the runtime environment, which executes the rules.

Another important benefit of a BRMS is that the lifecycle of the rules is separated from the lifecycle of the calling application or process.

As mentioned, rules normally change more frequently than the applications which call them. So the inefficient recompilation of the application every time a rule changes is not necessary anymore, because the new rules can be applied while the application runs.

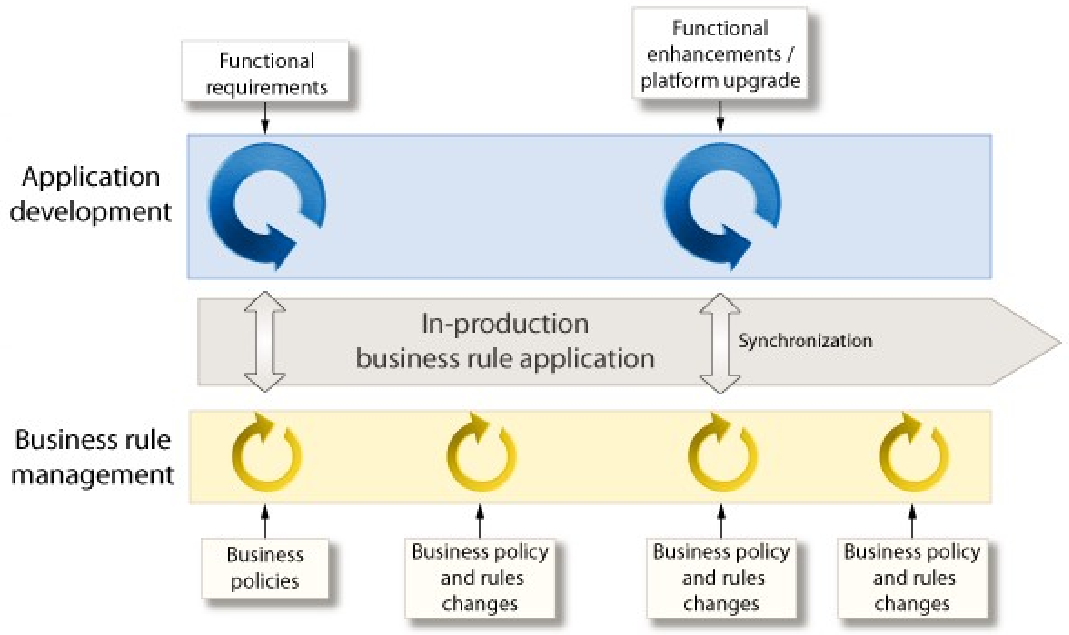

Functional requirements
Functional enhancements / platform upgrade
Application development
In-production business rule application
Synchronization
Business rule management
Business policies
Business policy and rules changes
Business policy and rules changes
Business policy and rules changes

Fig. 4.2: Business rules and the calling applications have different lifecycles [9]

Because of the reasons stated above a BRMS is a great help for a company to manage the decision logic of the applications and processes. Because decision logic exists in many industries, the usage of a BRMS can be leveraged in any industry.

Some use cases of a BRMS in the industry are [10]:

- Decision Services for claims processing and credit approval
- Data Validation and Fraud Detection for monitoring and alerting purposes
- Classification and Derivation for the scoring of customers or products
- Matching for the determination of suitable products or locations
- Calculation for the determination of costs or risks

4.1 Rules Management Lifecycle

The creation and management of business rules has it's own lifecycle and the quality of the way the steps in the lifecycle are handled by a BRMS differentiates it from other BRMS. This sub-chapter first gives an overview of the lifecycle of rules creation in a BRMS and then describes the key steps in more detail.

The general steps in the lifecycle of Rules Management are the following:

- Discovering and extracting the business rules from the applications and processes in the system.
- Defining a suitable vocabulary for the objects which are involved in the rule calling process, called "verbalization" - or taking an existing vocabulary.
- Constructing the skeleton for the later modeling of the rules using the given vocabulary
- Making the skeleton available for business users so that they can author the rules. Also IT developers are able to model rules using their tool.
- Testing the rules in a simulation and deploying them to the runtime environment of the BRMS if the tests were successful.
- Deploying the rules to the production system.
- Monitoring the rules if such a functionality is provided by the BRMS.
- Maintaining the rules and performing a rollback whenever it is necessary.

4.1.1 Verbalization

The objects which are involved in the rule calling process are normally defined in an XML Schema or Java Class. This Schema or Java Class is imported in the BRMS so that the objects are introduced to the rule engine, which will execute the rules based on these objects.

Since these objects are extracted from an XML Schema or a Java Class, their readability is very limited and they are not understandable to business users, who should be able to model the rules. Business users ideally model rules using a natural language with the expressions, they use themselves.

To enable this, a mapping must exist between the objects, which the rule engine works on, to custom expressions, which are defined by the user. Then the business rules can be modeled using the custom expressions and before the rules are executed by the rule engine the custom expressions will be mapped back to its original form. This mapping-process is called "Verbalization", which can be a mandatory step in a BRMS or be optional.

For example an object defined in a Java Class, which represents the credit score of a borrower in a loan request process, might initially have the following form:

`<Java Class>.<Member>`, like “`Borrower.creditScore`”. This would then have to be mapped to the custom expression “`the credit score`”, for example.

All the mapping procedure is part of building the skeleton of the business rules, which is the task of an IT developer or Business Analyst. The tasks and roles of the users involved in creating and managing business rules is explained later in this chapter.

4.1.2 Modeling

Business Rules can be modeled and grouped in different ways, depending on the possibilities offered by the BRMS.

If multiple business rules have a similar structure or have related content they can be grouped together in a graphical way, which simplifies their modeling. Below are the main constructs offered by most BRMS with which the rules can be modeled.

Business Rule:

A Business rule always has the following form:

```
if <Condition(s)> then <Action(s)> else <Action(s).
```

Example:

```
if the salary of the applicant is less than 1500 € and the
profession of the applicant is student

then set the score to '100'
```

Decision Table:

Each row of a decision table corresponds to a business rule. In this example the decision table defines whether a car rental is permitted to a customer or not, depending on the age of the customer. The first rule says that if the customer's age is between 18 and 21 then the rental is rejected. If it is between 21 and 25 then it is not.

Age of the Customer		Rental rejected
Minimum Age	Maximum Age	
18	21	true
21	25	false

Table 4.1: A Decision Table

As can be seen this decision table does not cover all ages a customer can have. If a customer's age is less than 18 or above 25 the rules in the decision table do not apply. To solve this either additional rows have to be added to the decision table or the decision table has to be combined with one or more business rules.

For example:

```
if the customer's age of the rental agreement is less than 18

then reject the rental agreement.
```

This business rule covers the cases where the customer's age is less than 18.

Decision Tree:

Another possibility to group rules is to use a decision tree.

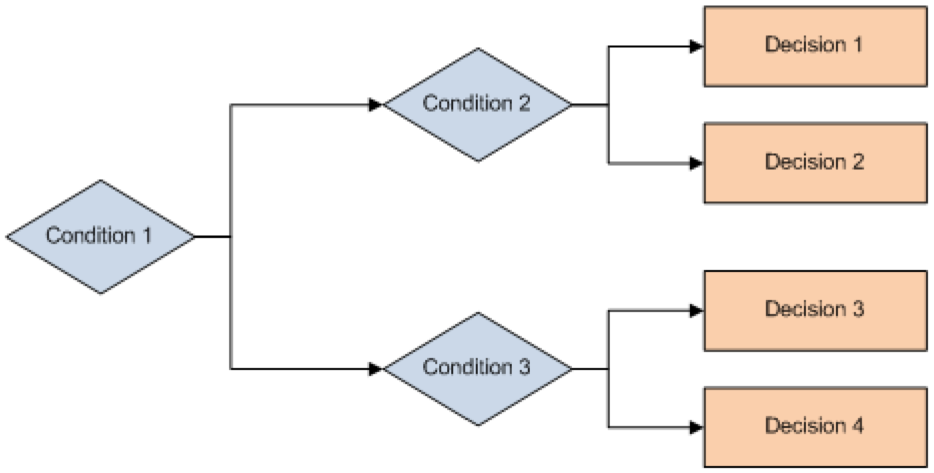


Fig. 4.3: A Decision Tree

In a decision tree a tree is created with branches and nodes. Each node represent a business rule and, depending on the result, a branch is taken which again leads to another node, until a leaf-node is reached. A leaf-node represents the result of the decision. A decision tree always starts with the evaluation of an initial business rule.

Ruleflow:

A BRMS can offer a graphical way to define a manual order for triggering modeled rules, which is called a "ruleflow".

A node in the ruleflow is a task why might consist of another ruleflow, a decision table, a decision tree or a simple business rule. Ruleflows are modeled using a graphical tool similar to BPMN (Business Process Modeling Notation). It has a start point and an end point and the logic is defined in between.

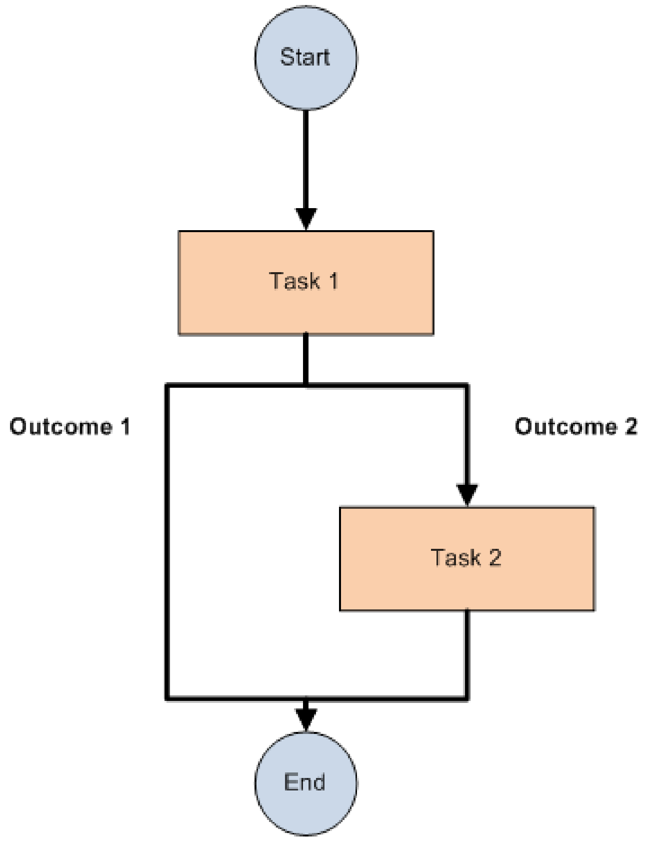


Fig. 4.4: A Ruleflow

Ruleset:

A ruleset represents a set of business rules. The purpose of a ruleset is to represent a certain policy or decision of the company, since most decisions are not that simple to be represented by a single business rule. For example during the process of checking the eligibility of a customer for a requested loan, multiple rules are involved, like the age verification and history of the customer at that bank. The ruleset for this decision can then for example be called "eligibility rules".

A ruleset can contain all kinds of rules, such as a number of simple business rules, decision tables, decision trees or ruleflows.

Rulesets are exposed to the calling application or business process as a black box and deliver the expected results of the decision to a certain business task. Therefore parameters have to be defined which represent the input and output data related to a ruleset. This concept is similar to the usage of parameters in a Java Method.

4.1.3 Testing

An important step in the Rule Management Lifecycle is to test whether the modeled rules show the expected results. This is normally done by executing the rules in a test environment with a set of sample data and expected results as input. The usage of a sheet to enter the input data is very suitable, therefore some BRMS offer the possibility to enter the input in an Excel Sheet using a familiar software, which is especially beneficial for business users.

Condition 1	Condition 2	...	Condition n	Expected Result
Value 1 for C1	Value 1 for C2	...	Value 1 for Cn	Result 1
Value 2 for C1	Value 2 for C2	...	Value 2 for Cn	Result 2
...	...	...	...	...
Value n for C1	Value n for C2	...	Value n for Cn	Result n

Table 4.2: Sheet with sample input data

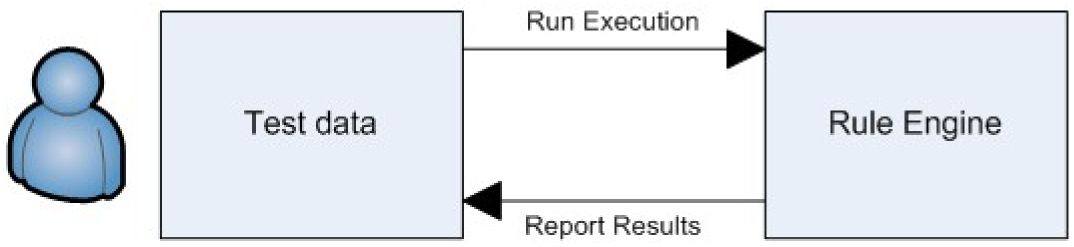


Fig. 4.5 : The Rule Engine executes the Test Data and sends the results back

4.1.4 Enabling Rules Management to Business Users

One of the benefits of a BRMS is that Business users who decide the rules in a company can author and manage the business rules themselves and do not have to delegate the task to the IT staff.

Since the initial creation of the business rules is mainly done by an IT developer or business analyst with technical background, a BRMS must enable the deployment of these rules to the environment for the business users. The quality of the environment is also a differentiating factor between different BRMS. The more intuitively the rules can be authored and managed by business users and the more functionality is provided to them, the better the BRMS is.

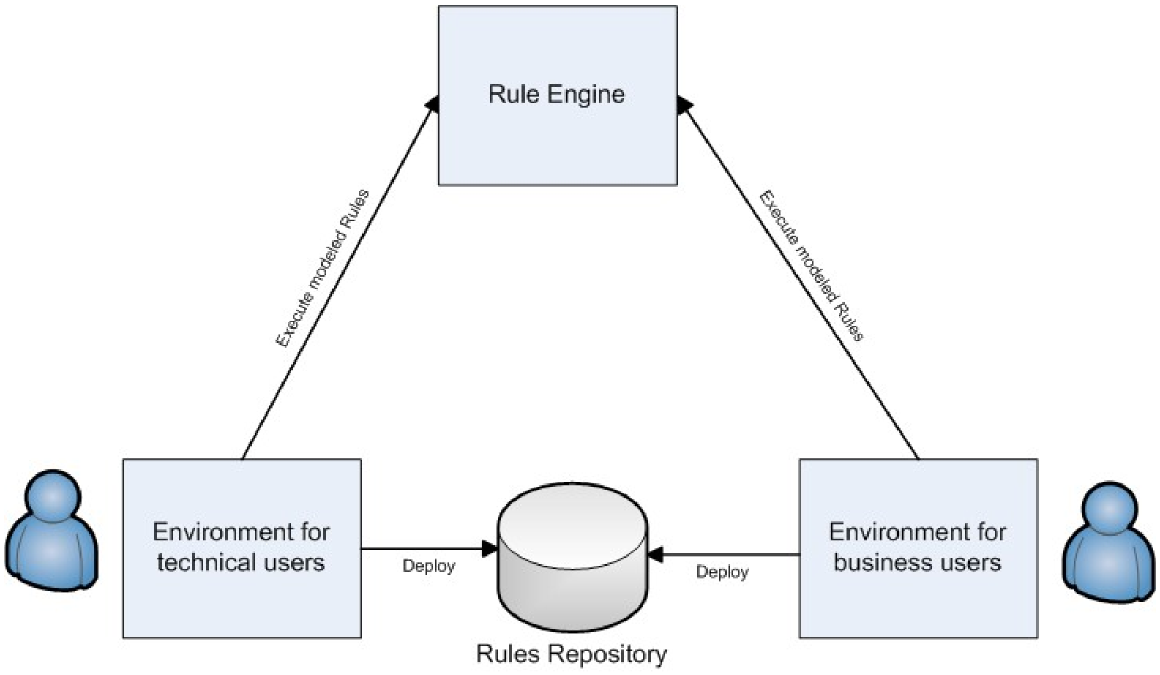


Fig. 4.6.: Scenario for enabling rules deployment and execution for technical and business environments

Ideally the rule engine is connected to both environments for the IT developers and Business Users, so that rules can be tested and deployed on both environments. The synchronization of the rules modeled in both environments is often accomplished by using a repository, where shared rules are stored.

4.2 The Roles involved in the usage of a BRMS

The development of a rule application using a BRMS involves multiple users representing different roles in the development and management.

While the number and depth of the tasks may vary in each BRMS, the following gives an example of the roles, which applies for most BRMS [9]:

Business Analyst:

A Business Analyst acts as the bridge between the IT and business units of the company. Ideally it understands both languages – on the one side the technical language of the IT developers and on the other side the precise intentions of the policies which are decided by the policy manager. So the main objective of a Business Analyst is to take the policies from the business side and translate it into a model which is acceptable for IT developers so that they can create the skeleton of the rules.

Because the Business Analyst understands the language of the business units he also has the task to define the vocabulary which is used in the business rules. The creation the first version of the rules which can later be maintained by for example the policy manager is also part of his job.

IT Developer:

The Business Analyst has created a vocabulary which is suitable for the definition of business rules. It's the task of the IT developer now to create the implementation of this vocabulary by mapping the vocabulary to the underlying objects which are modified and used by the Rule Engine (see below).

Another task of an IT developer is to make the creation of rules using the vocabulary available to the business users, which implies the synchronization of the rules which are modeled by the developer with the platform for business users (mostly a web-based platform).

There might be the situation that some rules can't be modeled by a business user, because their complexity requires the technical knowledge of an IT developer.

As described above business users can use simple if-then rules, decision tables, decision trees and maybe flow rules for modeling rules. But if for example loops are needed in a rule then it has to be written by an IT developer, because writing these so called "technical rules" is not suitable for business users.

Policy Manager:

Policy Managers are the ones who create the policies for the company. They are usually not familiar with any kind of technical language but are experts in working with excel sheets or text-documents. They normally collaborate with Business Analysts for the creation of rules in the environment which is intended for business users and they are responsible for updating the rules whenever new decisions are made.

Some BRMS offer the possibility to model the business rules using products which are familiar to policy managers, even more than a web-based environment. These products are normally Microsoft Excel for decision tables and Microsoft Word for simple business rules. The rules defined here have then to be synchronized with the repository of the BRMS.

Administrator:

The installation of the BRMS including the environments for IT developers, Business Analysts and Policy Managers is the job of the Administrator together with the configuration and maintenance of the environments.

Normally different roles with different user access is granted to certain users. The management of the user access is done by the Administrator.

Architect:

The optimization of the rulesets, which represent a certain business policy is done by the architect. Also the aspects for ensuring an optimal integration of the BRMS with it's rulesets into the enterprise application is part of his job.

4.3 The Rule Engine

The Rule Engine is the heart of every BRMS, because it is the engine which executes the modeled rules. It affects the speed with which rules deliver a result and the knowledge of the algorithms used in a rule engine can also help in modeling rules more effectively.

4.3.1 Design

A Rule Engine consists of a rule base, a set of facts and an inference engine.

Rule base:

In the rule base all business rules defined by the IT developers or business users are stored. Normally this is accomplished by using a repository.

Facts:

Facts represent the data which is being matched with the conditions of the rules to determine whether their conditions are satisfied. They are stored in the so called "working memory".

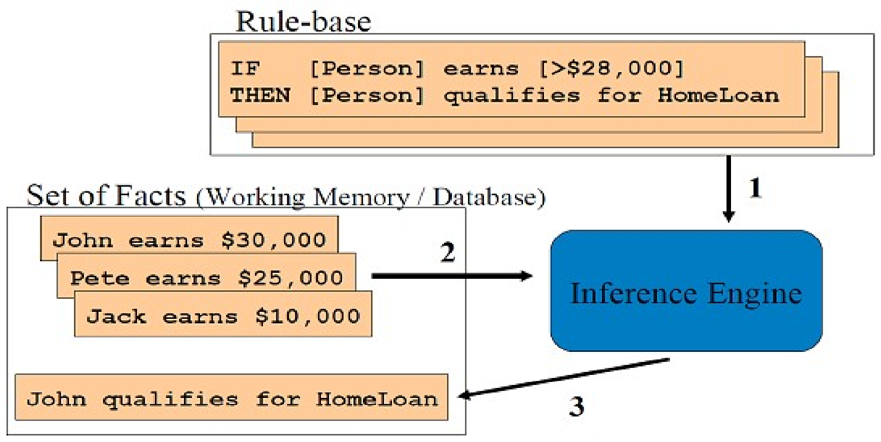

Rule-base
IF [Person] earns [>$28,000]
THEN [Person] qualifies for HomeLoan
1
Set of Facts (Working Memory / Database)
John earns $30,000
Pete earns $25,000
Jack earns $10,000
2
Inference Engine
John qualifies for HomeLoan
3

Fig. 4.7: The inference engine [11]

Inference Engine:

The inference engine consists of a Pattern Matcher and an Agenda.

The rules in the Rule Base and the Facts in the Working Memory are compared in the Pattern Matcher. The performance of a rule engine highly depends on the algorithm used in the Pattern Matcher.

In its simplest form each fact is compared with the rules one by one and the rules are fired if their conditions are met. But this becomes very slow with an increasing number of rules. Therefore there has been put research on creating algorithms which perform the task of Pattern Matching much faster, e.g. by taking into account the similarity in the structure of some rules. The most popular and common algorithm for pattern matching is the Rete-Algorithm, which will be explained in the next chapter.

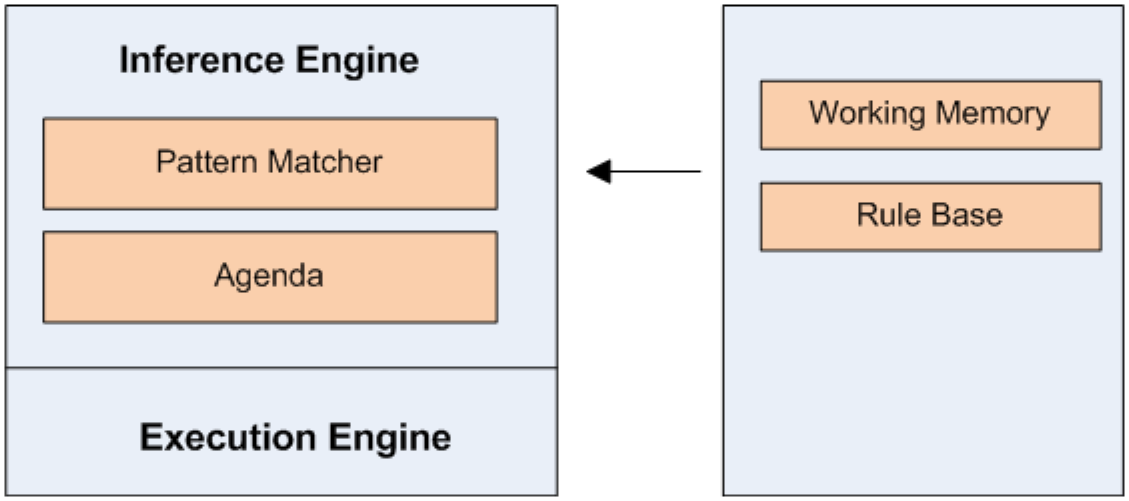


Fig. 4.8 : The design of a rule engine

The Pattern Matcher puts the rules whose conditions are satisfied in a queue for execution, called Agenda.

After comparing the rules with the facts there might be the case that the conditions of more than one rule are satisfied. The determination of the order in which the rules are fired is the task of the Agenda, which needs a "conflict resolution strategy".

<u>**Conflict Resolution Strategies:**</u>

There are usually four conflict resolution strategies, which in some BRMS can also be influenced by the modeler in the design time [12]:

– **Specifity:** If more than one rule is eligible to be fired, the one with the most specific condition is chosen. E.g. "customer age > 18" is fired after "customer age > 18 and customer score > 500".

- **Priority:** The modeler of the rules can give a specific priority to each rule, so that the rules with higher priority will be executed first.
- **Recency:** The rule whose conditions use the most recently added or modified facts is fired.
- **Refractoriness:** If the conditions a rule are satisfied by the facts, but previously the same rule has been satisfied by the same facts, the rule is ignored. This prevents the engine from entering infinite loops.

Finally the execution engine is responsible for executing the rules. The execution of a rule can modify existing facts in the working memory which might change the status of the rules in the Agenda.

Some new rules might be put in the agenda, because their conditions are satisfied with the new facts, and some rules might be removed from the agenda, because their conditions are not satisfied anymore.

4.3.2 Pattern Matching algorithms:

The execution speed of the rule engine depends on the speed with which the facts in the working memory are matched against the rules in the rule base. Therefore the rule engine uses algorithms for that task.

The algorithms have different strengths and their performance depends on the structure of the rulesets, which they evaluate. The most popular algorithm which is used in most Rule Engines ist the "Rete"-Algorithm. Another algorithm, which is often used, is the "Sequential" Algorithm.

If the BRMS offers the possibility to choose an algorithm for a specific ruleset, the modeler should know the strengths of each algorithm to ensure an optimal performance.

As mentioned, the execution of a rule can result in the modification of the existing facts in the working memory or the addition of new facts. This can modify the status of the rules which are in the agenda – some new rules might be put into the agenda and some might be removed.

If no specific algorithm is used by the rule engine, all facts are matched with all rules after each iteration, which is very inefficient.

<u>Rete-Algorithm</u> [9]**<u>:</u>**

The goal of the Rete-Algorithm is to evaluate after each iteration only those rules, whose conditions might have been modified. Therefore rules, whose facts have not been affected by the current iteration, are ignored.

Because normally the number of facts which are modified in an iteration is relatively low and the number of rules in a ruleset is relatively high, the usage of the Rete-Algorithm results in a significant performance gain.

The Rete-Algorithm creates a network of nodes, where each node represents a part of the condition of a rule.

The type of nodes are:

– Kind node: Each kind node represents a type of the facts in the working memory. For example the "age"-fact is a kind node or the "score"-fact is also a kind node.

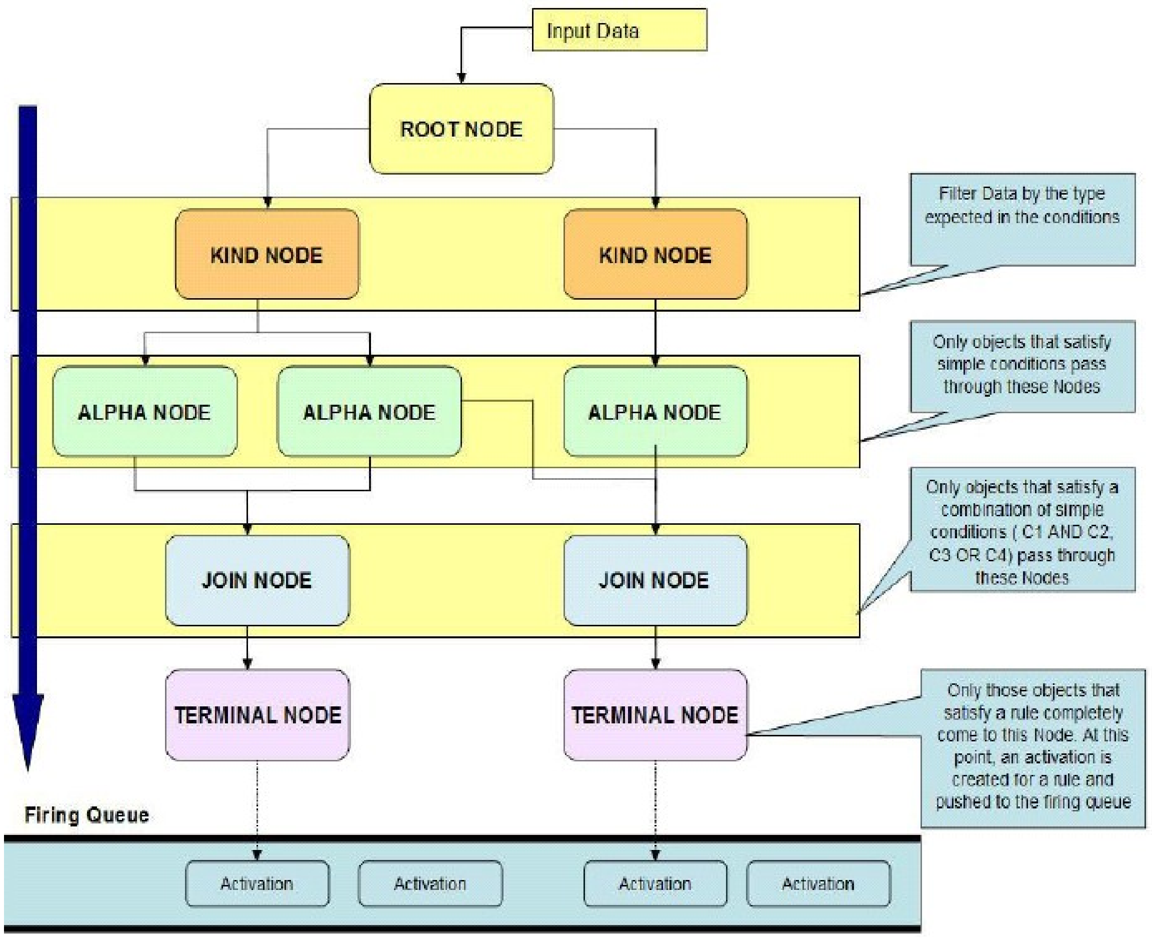
– Alpha node: An alpha node is one single condition of a rule. E.g. "age > 18".

– Join node / Beta node: A beta node is the conjunction of two alpha nodes. E.g.: the two alpha node "age >18" and "score > 500" can be joined to the beta node "age>18 and score > 500"

– Leaf node / Terminal node: If a leaf node is reached, the complete conditions of a rule are satisfied and the rule is put in the agenda for firing.

Each node has a memory with a reference to the facts which are relevant for the node, so that only rules whose conditions use modified or newly added facts are evaluated.

As a result, the drawback of the Rete-Algorithm is the high memory it needs.



Input Data
ROOT NODE
KIND NODE
KIND NODE
Filter Data by the type expected in the conditions
ALPHA NODE
ALPHA NODE
ALPHA NODE
Only objects that satisfy simple conditions pass through these Nodes
JOIN NODE
JOIN NODE
Only objects that satisfy a combination of simple conditions (C1 AND C2, C3 OR C4) pass through these Nodes
TERMINAL NODE
TERMINAL NODE
Only those objects that satisfy a rule completely come to this Node. At this point, an activation is created for a rule and pushed to the firing queue
Firing Queue
Activation
Activation
Activation
Activation

Fig. 4.9: The network built by the Rete Algorithm [14]

Sequential Algorithm [9]**:**

The sequential algorithm does not work with an agenda. Instead it takes the input data and matches it with the conditions of the rules in the rule base. Whenever a matching is found, the corresponding rule is fired immediately and can generate output data or modify facts.

The Sequential Algorithm is suitable for a ruleset with a high number of different rules, which have different conditions, e.g. for validation purposes.

The Rete Algorithm on the other hand is suitable for all other rulesets, especially those who have a similar structure in the condition part.

4.4 Rules Managed in BPM tool vs. Rules Managed in BRMS

Most tools for Business Process Management offer the functionality to define simple business rules while designing a process. The functionality of modeling rules in these tools does not offer the complexity and management capabilities as a BRMS, but for simple use cases, like branching decisions, the functionality of a BPM tool is more reasonable than a full BRMS.

Another use case for using the BPM tool is if lifecycle of the rules modeled in the process is the same as the process lifecycle.

Below is a table with comparisons that help in deciding whether rules should be modeled directly in the BPM tool or should be sourced out to a BRMS [15].

Use a BRMS when	**Use the BPM tool when**
The rule management life cycle must be independent of the BPM life cycle (implies that the frequency of change is different between a decision and the process)	The rules belong to the life cycle of the process
Rules are authored and maintained by business users	Rules are authored and maintained by the same user that owns the process model
Decisions contain many rules	Decisions contain very few rules
Decisions require independent testing and simulation by business users	Decisions are tested and simulated at the same time as the process
Decision require inference or contain complex data structures	Data used for decisions is limited to routing and basic evaluation
Rule content must be shared between multiple applications(Java, COBOL, .NET)	Rules belong to one application or process

Table 4.3.: Use Cases for using rules in BPM tool and BRMS

5 IBM BRMS and BPM Solutions

IBM's BRMS is called WebSphere ILOG JRules and has it's origin in IBM's acquisition of the french Company ILOG in 2006. ILOG's BRMS JRules was one of the market leaders in BRMS and there was already a lot of cooperation between IBM's middleware products from the WebSphere brand and JRules.

The intention of this chapter is to give a detailed description of the functionality of JRules and how it is integrated in IBM's ecosystem.

5.1 WebSphere

WebSphere is the brand name of IBM's middleware software products for e-business. It's general purpose is to enable the integration and collaboration of enterprise applications on an SOA platform.

While the list of middleware products, which are part of the WebSphere brand, is long, here is a selection of key products, divided by following categories [16]:

- **Application Servers**: WebSphere Application Server, WebSphere Application Server Community Edition (free version)
- **Business Integration**: WebSphere MQ, WebSphere Message Broker, WebSphere Enterprise Service Bus
- **Process Integration**: WebSphere Process Server, WebSphere Integration Developer, WebSphere Business Events

The products which are relevant in the context of this diploma are the BPM tools, which enable process integration, because the services of a BRMS will be part of the Business Processes, which run on the Process Engine. Therefore the following 3 products will be described in more detail:

- **WebSphere Integration Developer**: The BPM tool, where the Business Process is modeled and built.
- **WebSphere Process Server**: The engine, on which the built processes are deployed and executed.

– **WebSphere Application Server**: The main middleware component, which connects different applications and where the Process Server is deployed on.

Because JRules is also a middleware component it also belongs to the WebSphere brand, but will be described in a separate chapter.

5.1.1 WebSphere Application Server

The WebSphere Application Server (WAS) is IBM's main Java EE certified Application Server and the flagship of the WebSphere brand. As an application server, it acts as the middleware between clients and back-end systems, such as databases. All enterprise applications are deployed on this server, including WebSphere Process Server and also the Rule Engine of ILOG JRules, which communicate through open standards. Considering the SOA reference architecture, WAS provides the Business Application Services, which is again highlighted in the following picture:

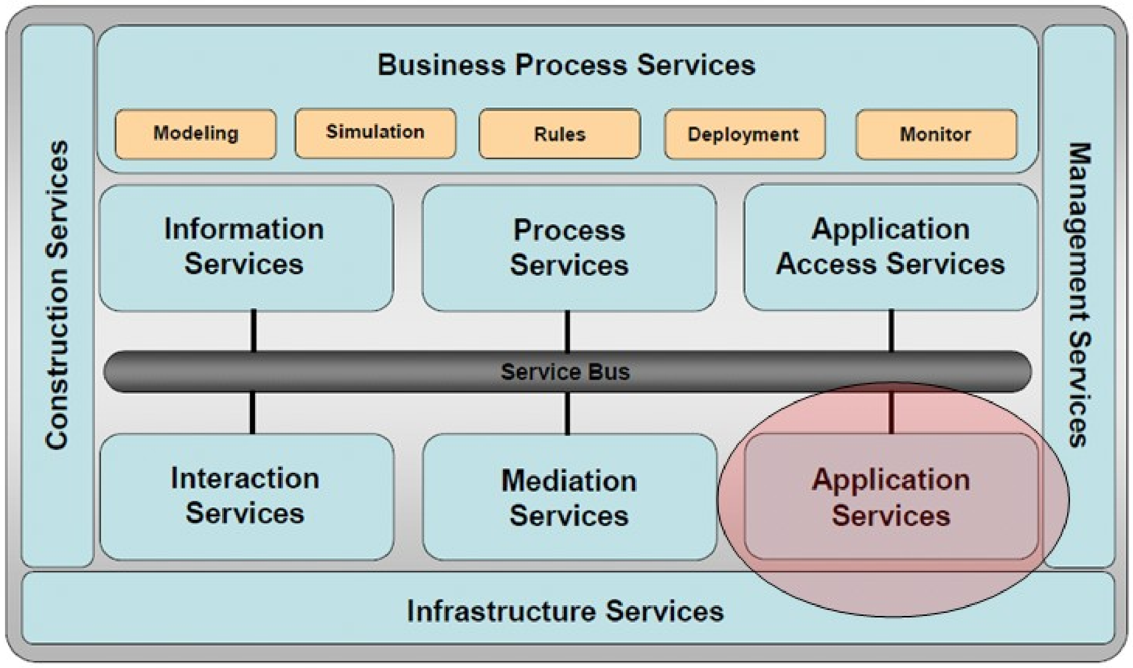


Fig. 5.1.: WAS provides the Application Services in a SOA

As a key SOA building block it provides mainly the following functions [17]:

– Reusable application services can be built and deployed quickly and easily

– Provides a secure, scalable and highly available environment to run services
– Software assets can be connected and extend their reach
– Applications can be managed effortlessly

Please note that JRules is by default shipped with the WebSphere Application Server Community Edition (WebSphere CE). This is a lightweight, open source application server which is suitable for departmental applications and testing purposes.

5.1.2 WebSphere Process Server

The WebSphere Process Server (WPS) is the execution environment for automated end-to-end business processes. It contains and extends the WebSphere Enterprise Service Bus, which orchestrates the services in a defined order using "Business Process Execution Language" (BPEL) and provides a platform for a Service Oriented Architecture. IBM gives the following definition of WPS:

> "WebSphere Process Server allows the deployment of standards-based business integration applications in a service-oriented architecture (SOA), which takes everyday business applications and breaks them down into individual business functions and processes, rendering them as services. Based on the robust Java EE infrastructure and associated platform services provided by WebSphere Application Server, WebSphere Process Server can help you meet current business integration challenges. This includes, but is not limited to, business process automation.
>
> WebSphere Process Server enables the deployment of processes that span people, systems, applications, tasks, rules, and the interactions among them. It supports both long-running and short-running business processes, providing transaction rollback-like functionality for loosely coupled business processes." [18]

The main difference between WebSphere ESB and WebSphere Process Server is that WPS uses BPEL for execution, while WebSphere ESB mainly acts as a message broker for communication between a service consumer and a service provider. WPS is deployed on an application server, normally the WAS, and offers an administration environment with rich configuration options.

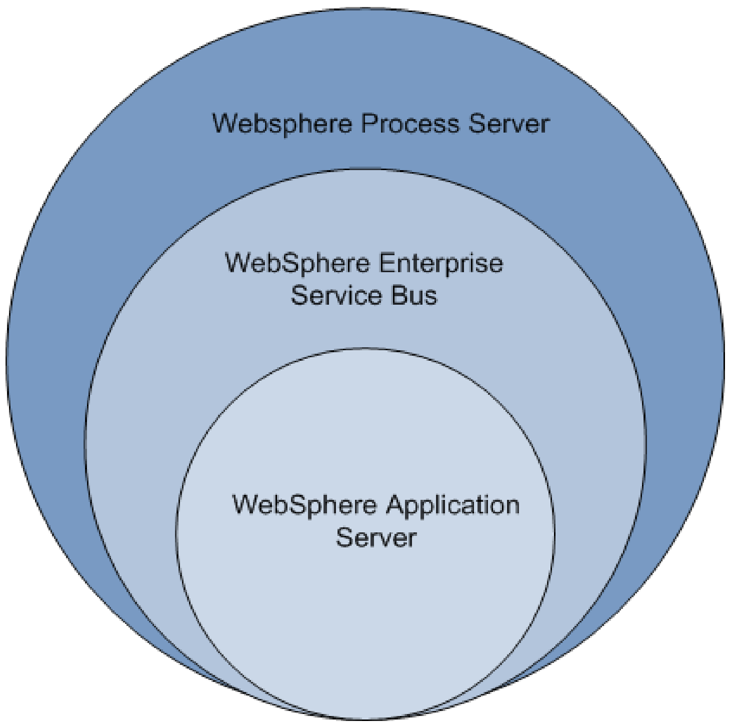

Websphere Process Server
WebSphere Enterprise Service Bus
WebSphere Application Server

Fig. 5.2: WAS is the base for both WPS and WESB, WPS extends WESB

5.1.3 WebSphere Integration Developer

The development of SOA-based applications, which run on the WebSphere Process Server or WebSphere ESB is done using WebSphere Integration Developer (WID). The WID is an eclipse-based authoring tool which enables the creation of business processes by a non-technical user through the assembling of services.

The assembling is made using the principles of the Service Component Architecture, which is explained in the following [19].

Service Component Architecture:

In WID an SOA is modeled using the artifacts proposed in the specifications of the Service Component Architecture (SCA).

In the SCA programming model a business process is composed of service components. A service component represents a service which can be consumed by other components or consumes services itself. There are multiple options available for the implementation of a component, as illustrated in the picture below.

The main advantage of SCA is that the business logic is decoupled from the implementation details, so that the business process can be modeled by the integration developer and the implementation is done in a separate step by IT developers.

If the interfaces of a component stay the same, the implementation of the component can be replaced without any affection of they way it is accessed. For example a component representing an approval task which is done by a human can be replaced by a business rule, while the interface remains the same. While the functionality of a component is offered to other components through an interface, the component itself consumes other services through a reference.

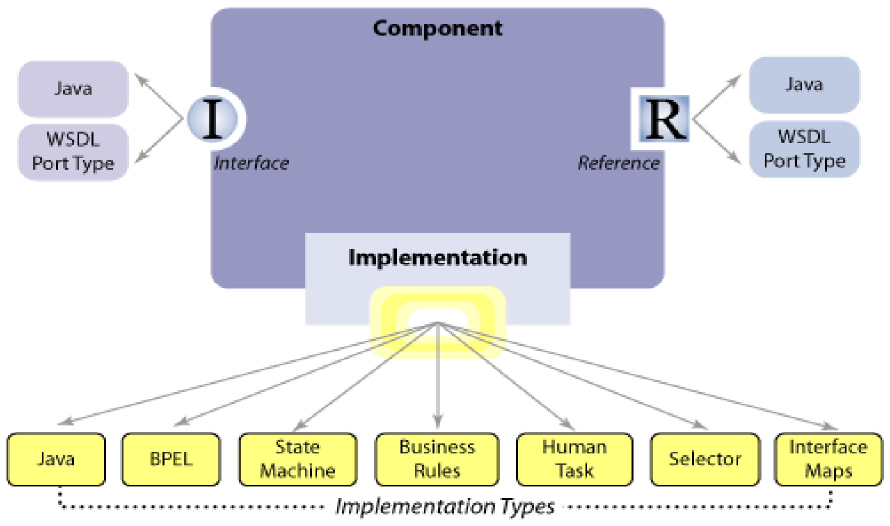


Fig. 5.3: Design of a Component in an SCA [19]

An interface specifies the input data which the service expects and the expected output data which is returned by the service. It can be specified as either a Java Interface or WSDL port type. A reference defines the output data which will be sent by the component and the input data which the component will receive. A reference is then connected to the interface of another component.

Using this concept, in the so called "Assembly-Diagram" the components can be connected graphically.

SCA module:

Related components are grouped together in an SCA module, which is a unit of deployment. The components inside a module communicate using interfaces and references, as described above, but have first no access to components outside of the module. This is accomplished by using "Imports" and "Exports". Imports are used to consume the services of other SCA modules. Once an Import has been defined, each component inside the module can consume the Services exposed by the other module through the Import. The Export of a module on the other hand exposes a particular service of the module to other modules.

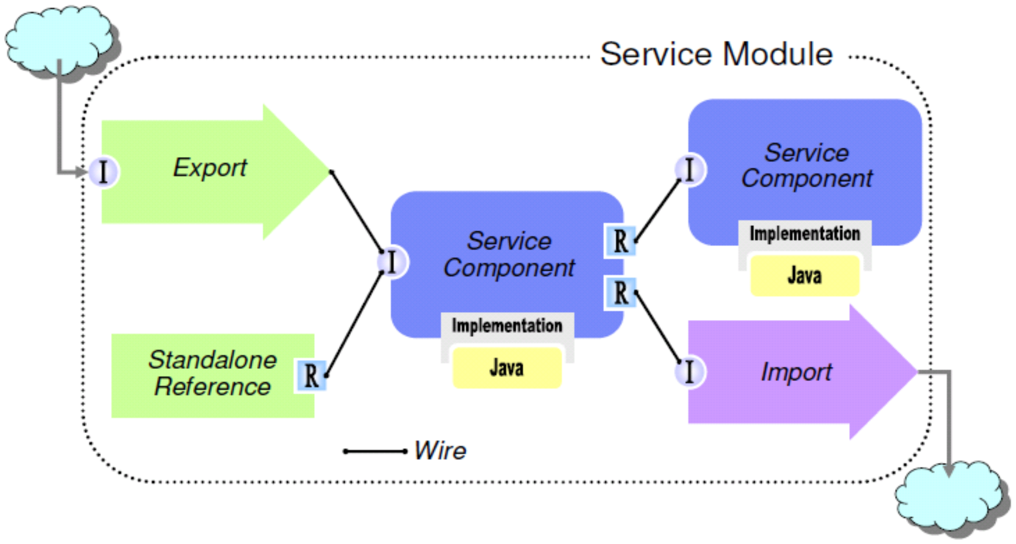


Fig.5.4: Design of a Service Module in SCA [19]

There are two types of modules: **Business integration modules** (or just "modules") and **mediation modules**.

The purpose of a Business integration module is to support a business process through the services offered by it's components. The mediation module acts like a gateway for external services, which is typical in an ESB Architecture. Inside the mediation module there is a component called "mediation flow", which has the task of data mapping or logging mechanism.

Both module types can be deployed to the Process Server in an EAR file.

5.2 WebSphere ILOG JRules

ILOG, which was acquired by IBM in 2006, is a french company which develops enterprise software applications in four categories [20]:

- Supply Chain Solutions
- Business Rules Management Systems
- Visualization
- Optimization

The focus in this thesis will be on ILOG's BRMS called "JRules" (Version 7.1). It is one of the market leading BRMS due to a complete coverage of the functionality of an advanced BRMS. It provides graphical tools for business users with which they can author, test and manage business rules intuitively.

JRules comprises a set of different tools which are tailored to the skills of the persons who are involved in the management of a BRMS. An overview of the tools together with the corresponding roles in creating a rule application is given in the following picture:

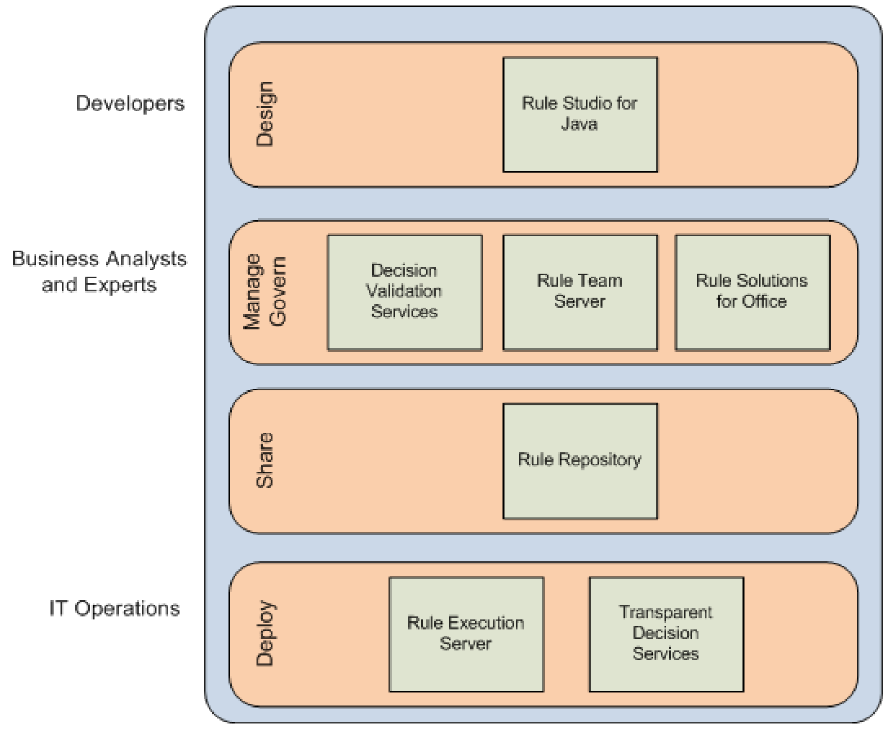


Fig.5.5: Design of JRules

The main environments for creating and managing rules are Rule Studio, Rule Team Server and Rule Execution Server, which will be first introduced briefly. The other components which can be seen in the picture will be explained while going step by step through the lifecycle of rule management.

Rule Studio:

Rule Studio is an eclipse-based IDE for IT developers to create and manage business rules. The initial creation of the skeleton of the rules, which the business users later work on is done here. Deployment of Rules to the Execution Environment can also be done from here.

Rule Team Server:

This is the web-based environment for business users to author, review and deploy business rules independently from IT. The authoring and management of rules (amongst other features which will be focused on later) is performed using graphical tools in the Web-Browser - the user won't see any code here.

Rule Execution Server:

The Rule Execution Server is a web-based environment where the finished rule projects, in JRules called "RuleApps", are deployed. Both the IT developers modeling rules in Rule Studio and the Policy Manager modeling rules in Rule Team Server deploy their rules to the Rule Execution Server. It is J2SE- and Java EE-compliant and is deployed to an Application Server. Statistics on the rules fired in an application e.g. for auditing purposes can be fetched here, amongst many other features useful for the management of deployed RuleApps.

5.2.1 Verbalization

The rule engine needs to know which objects to work on. These objects must first of all be determined and then introduced to the Rule Project. In JRules these Objects are called the "Execution Object Model" (XOM).

Depending on whether the calling application is a Java Application or a Process using Business Objects the XOM is either a Java Class or an XML Schema. The rule engine can manipulate the objects or execute the methods provided in the XOM. But the rules aren't modeled against the XOM, instead they reference the BOM, which stands for "Business Object Model".

Because the XOM does not necessarily have to be readable for humans it is unsuitable for defining rules, since readability is an essential for business rules. The objects in the BOM are expressed using natural language. At runtime, rules that were written against the BOM are mapped to a technical language and run against the XOM.

Every BOM element must therefore have a corresponding XOM element. If the mapping from BOM to XOM is one by one, i.e. every BOM-element has a corresponding XOM-element, the BOM can be generated automatically in JRules using the option "Generate BOM from XOM". However, if not, some mappings have to be defined manually by the user.

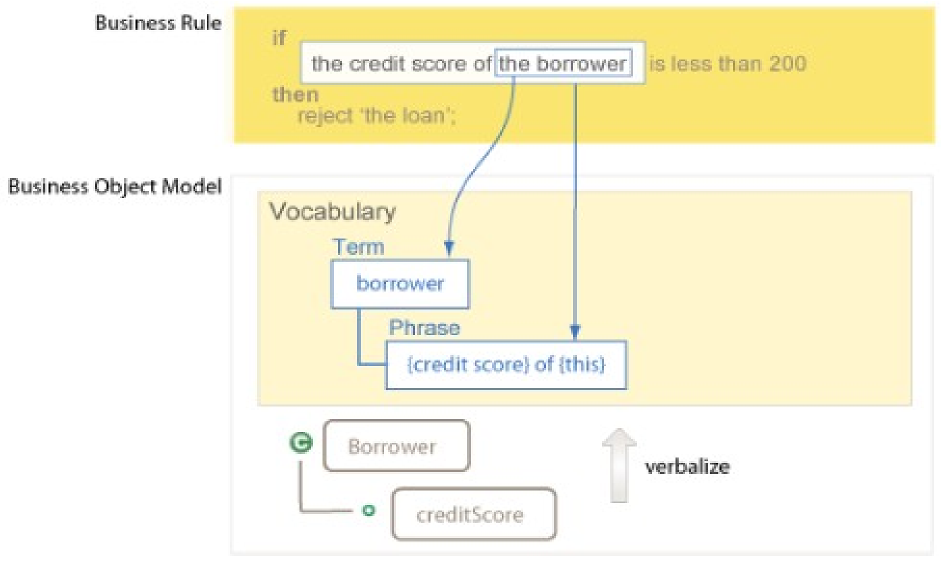


Fig. 5.6: Verbalization through a XOM to BOM mapping [9]

A manual mapping of BOM to XOM is for example needed when some terms, which would facilitate the creation of rules and enhance their readability, don't exist as XOM. One example would be that the XOM has an Object which represents the date of a customer's birth, but having the age of the customer in years would be more suitable for rule creation.

JRules then offers the possibility of calculating the age from the birth date using Java Code and then adding the age of the customer in the BOM model, so that it can be used for rule creation.

Rule Languages used in JRules:

The rule language which the rule engine of JRules expects is the "ILOG Rule Language" (IRL), which has a Java-like syntax.

An IT Developer in Rule Studio has the choice to use the IRL directly to model the rules or to use the structured English syntax using the BOM, which is nearly natural language.

This readable language is the so called "Business Action Language" (BAL). The BAL is the language which the business users will use to author and modify the rules in the Rule Team Server.

Here is an example of a rule modeled in BAL:

```
if the customer's age of 'the current rental agreement' is less than 18
then
reject 'the current rental agreement' ;
```

Because the BAL is not understood by the rule engine it is only used for the modeling of rules.
When these rules are compiled they are transformed into the IRL for execution.
Here is the same rule modeled in IRL:

```
package eligibility {
   rule UnderAge {
      property ilog.rules.business_name = "UnderAge";
      property ilog.rules.package_name = "eligibility";
      property status = "new";
      when {
         carrental.RentalAgreement() from rental;
         evaluate (rental.getCustomerAge() < 18);
      } then {
            rental.reject();
        }
    }
}
```

The rule is called `"UnderAge"` and resides in the package `"eligibility"`.

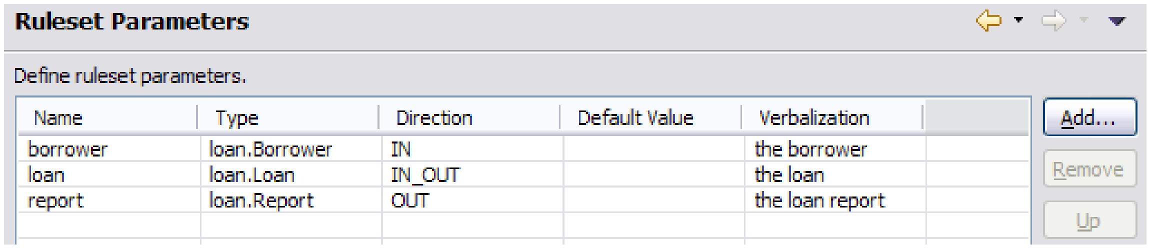
As can be seen the `"rental"`-Object in the IRL example is taken from the XOM which the Rule Engine works against and is called `"the current rental agreement"` in the BOM, which the BAL-rules are modeled against. The important process of creating the BOM-Vocabulary from the XOM is called "Verbalization".

An IT developer who models rules in Rule Studio normally uses the BAL. But they also sometimes use the IRL for the modeling of technical rules, which can't be modeled using BAL. These are complex rules which might for example contain loops and cannot be modeled by business users in the Rule Team Server.

Ruleset Parameters:

The definition of the XOM which is used by the Rule Engine makes it possible to define the input and output parameters of the ruleset which will later be the interface of the exposed service.

XOM-Variables can be defined as Input parameters (`IN`), as output parameters (`OUT`), and as parameters which are used as input and will be returned as part of the result (`IN_OUT`). These `IN_OUT` parameters can be modified and sent back to the service requester.


Ruleset Parameters

Define ruleset parameters.

Name	Type	Direction	Default Value	Verbalization
borrower	loan.Borrower	IN		the borrower
loan	loan.Loan	IN_OUT		the loan
report	loan.Report	OUT		the loan report

Fig. 5.7: When the ruleset is exposed as a service, the ruleset parameters define the interface

In this example the ruleset will calculate if a certain loan is given to a customer or not. The customer itself together with its attributes is the input to the ruleset. The requested loan also serves as input but will also be returned to the requestor and a report which contains a positive or negative response is the return value.

In the “Type” column the XOM can be seen that is used for this purpose. It is a Java-Class called “loan”, which contains the variables “Borrower”, “Loan” and “Report”.

5.2.2 Modeling

Business Rules:

As mentioned, Business Rules in JRules are modeled against the BOM.

```
if
the name of customer is "John Smith"
then
set the status of report to "Rejected" ;
set the reasons of report to "Bad Client" ;
```

Here the BOM variables are “customer” and “report”.
It is also possible to define temporarily used variables, which are called “definitions”.

Technical Rules:

As an alternative to business rules, which are modeled against BAL, technical rules support constructs which are not available in BAL, such as loops. Therefore they are written by IT developers in the technical language IRL.

Naturally loops only appear in the action-part of a rule and not in the condition part.

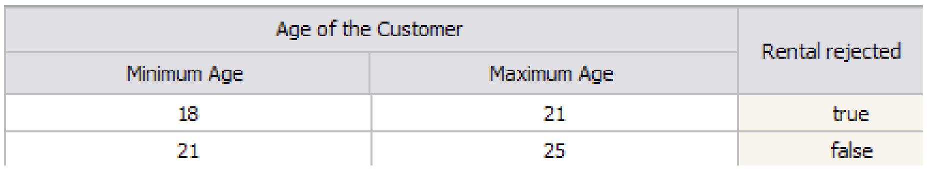
Here is an example of a technical rule:

```
then {
   int ?i = 1;
   for ( ?i = 1; ?i <= 3; ?i++ ) {
         System.out.println(" i =" + ?i) ;
   }
}
```

In this simple for-loop the action-part of this rule would generate the output “i=1 i=2 i=3”, which could be displayed to the rule caller.

Decision Table:

JRules provides the functionality of grouping rules with a similar structure to a decision table. When creating a decision table the condition columns and action columns are defined. Horizontal conditions are also supported, which enable a second dimension for possible actions. In the picture above "Age of the Customer" is a horizontal condition and "Minimum Age" and "Maximum Age" are vertical conditions.



Age of the Customer		Rental rejected
Minimum Age	Maximum Age	
18	21	true
21	25	false

Fig. 5.8: A decision table in JRules

If there are any overlaps or gaps in the condition rows of the decision table, JRules indicates this by displaying an error sign at the affected rows.

Decision Tree:

In order to be able to manage a set of business rules, which do not have a similar structure but are part of a business decision, they can be managed using a decision tree. A decision tree in JRules displays the relation between rules using three components:

- Diamond shaped nodes, representing a condition of a business rule
- Branches, representing possible values of a condition
- Action nodes, representing the actions which are executed when the corresponding condition is met

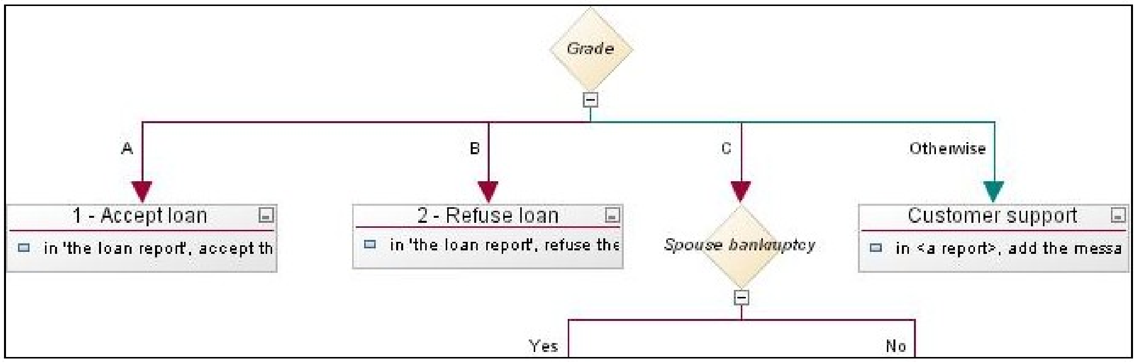

Grade
A
B
C
Otherwise
1 - Accept loan
in 'the loan report', accept th
2 - Refuse loan
in 'the loan report', refuse the
Spouse bankruptcy
Customer support
in <a report>, add the messa
Yes
No

Fig. 5.9.: A decision tree in JRules [9]

Ruleflow:

To define the order in which the modeled rules will be executed, a ruleflow is used. A ruleflow is made up of a sequence of "rule tasks", where each rule task is a business rule or a grouping of business rules, like a decision table.

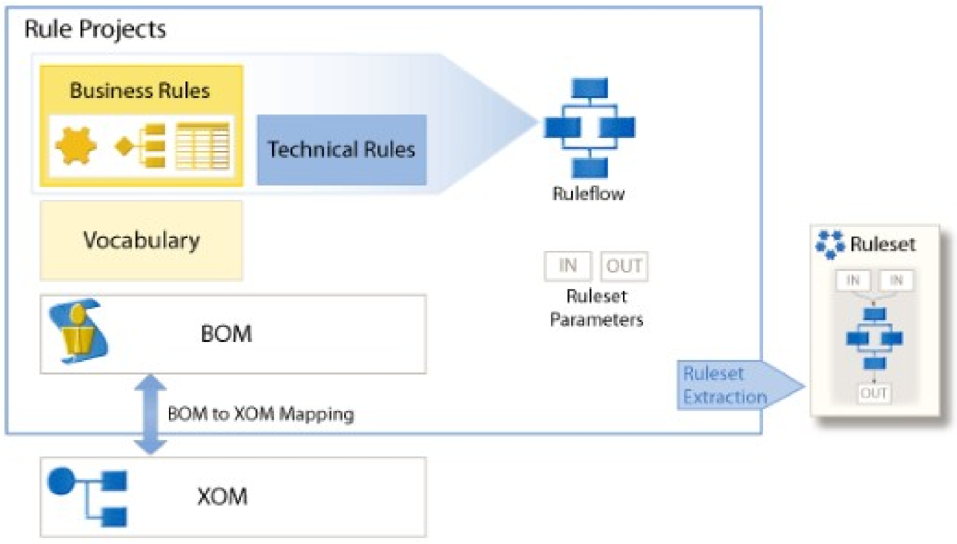


Fig. 5.10: A ruleflow consists of business rules and technical rules [9]

In this example the rule flow processes a loan request by a customer. It first evaluates the customer's eligibility for the loan and then calculates the pricing.

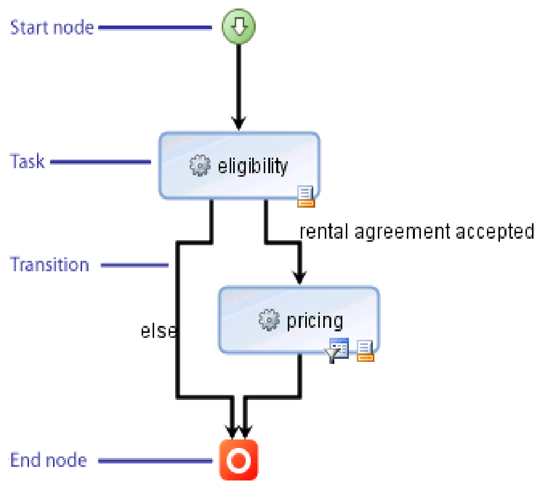


Fig. 5.11: A ruleflow in JRules [9]

The processing of the input parameters of the ruleset starts at the Start node. The output parameters are calculated in the rule tasks and returned at the End node.

In JRules a ruleflow can be the rule task of another rule flow, which allows a nested execution.

5.2.3 Testing

There are generally two options for Rule Testing:

- Providing a value for the input parameters and checking the result in the Run/Debug Configuration menu.
- Running a Test Scenario with a sequence of input values using the "Decision Validation Services".

Run/Debug Configurations:

A ruleset always includes input- and output-parameters, which it expects. Using the Run/Debug Configuration Menu in Eclipse a sample input can be provided and the ruleset can be executed with it. If it does not show the expected behaviour, the debug mode enables a detailed testing of the artifacts in the ruleset using trigger points, like the debugging of a java application.

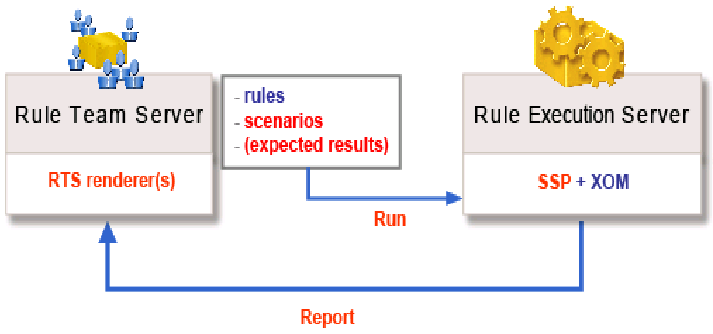


Fig. 5.12: Test Scenarios can be created on RTS and executed on RES through SSP [9]

<u>Decision Validation Services:</u>

The purpose of Decision Validation Services is to provide powerful rule-testing features for both IT and business users. The correctness and impact of changes to rules can be tested using scenarios with multiple variations of input parameters. Scenarios are modeled using for example an excel sheet.

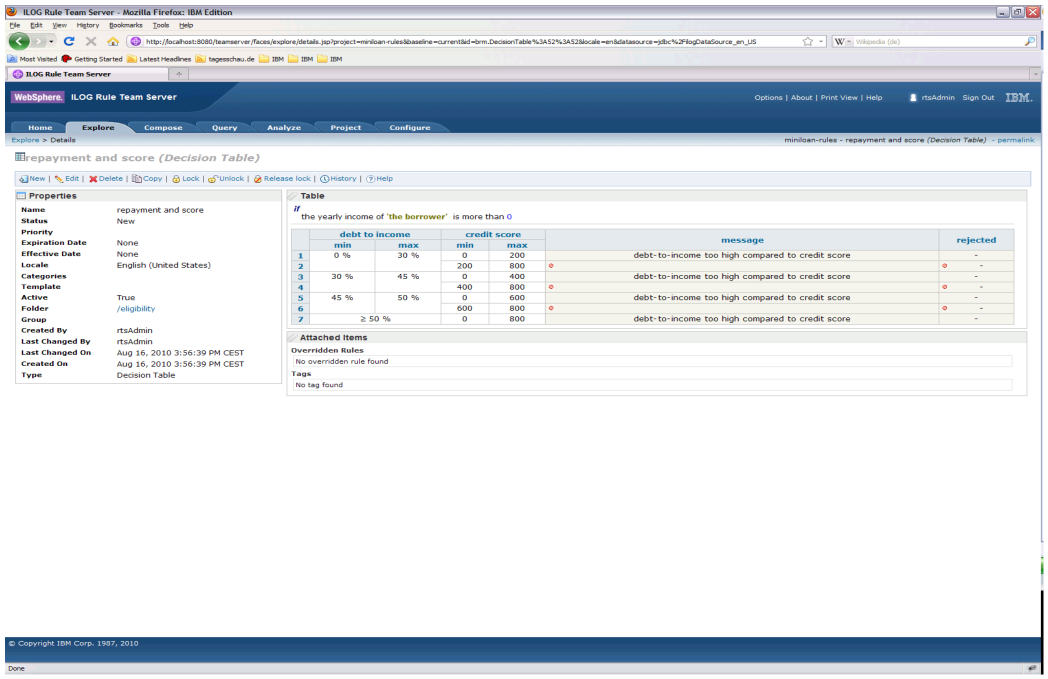
The execution of the test-scenarios is done by an application called "Scenario Service Provider" (SSP), which has to be deployed on the application server together with the XOM of the rule project. This provides the usage of this functionality outside in the Rule Team Server for Business Users.

The output of the test results is generated as an HTML report.

5.2.4 Rules Management for Business Users

Business users manage and author rules in a web-based environment called "Rule Team Server" (RTS).

Rule Team Server allows business users to view, author, manage,test and query business rules. From Rule Studio the rule project is deployed to Rule Team Server and then the work of the business users is periodically synchronized with the copy in Rule Studio.


ILOG Rule Team Server - Mozilla Firefox: IBM Edition
File Edit View History Bookmarks Tools Help
http://localhost:8080/teamserver/faces/explore/details.jsp?project=miniloan-rules&baseline=current&id=brm.DecisionTable%3A52%3A52&locale=en&datasource=jdbc%2FilogDataSource_en_US
Most Visited Getting Started Latest Headlines tagesschau.de IBM IBM IBM
ILOG Rule Team Server
WebSphere. ILOG Rule Team Server
Options | About | Print View | Help rtsAdmin Sign Out IBM.
Home Explore Compose Query Analyze Project Configure
Explore > Details
miniloan-rules - repayment and score (Decision Table) - permalink
New | Edit | Delete | Copy | Lock | Unlock | Release lock | History | Help
© Copyright IBM Corp. 1987, 2010
Done

repayment and score *(Decision Table)*

Properties

Name	repayment and score
Status	New
Priority	
Expiration Date	None
Effective Date	None
Locale	English (United States)
Categories	
Template	
Active	True
Folder	/eligibility
Group	
Created By	rtsAdmin
Last Changed By	rtsAdmin
Last Changed On	Aug 16, 2010 3:56:39 PM CEST
Created On	Aug 16, 2010 3:56:39 PM CEST
Type	Decision Table

Table

if
the yearly income of 'the borrower' is more than 0

	debt to income		credit score		message	rejected
	min	max	min	max		
1	0 %	30 %	0	200	debt-to-income too high compared to credit score	-
2			200	800		-
3	30 %	45 %	0	400	debt-to-income too high compared to credit score	-
4			400	800		-
5	45 %	50 %	0	600	debt-to-income too high compared to credit score	-
6			600	800		-
7	≥ 50 %		0	800	debt-to-income too high compared to credit score	-

Attached Items

Overridden Rules
No overridden rule found

Tags
No tag found

Fig. 5.13: Environment of Rule Team Server

<u>Rule Solutions for Office:</u>

JRules also provides the possibility to author rules in software products which are familiar to business users, namely Microsoft Excel for Decision Tables and Microsoft Word for If-then rules.

The rules can then be authored offline by the policy manager for example and then kept synchronized with Rule Team Server.

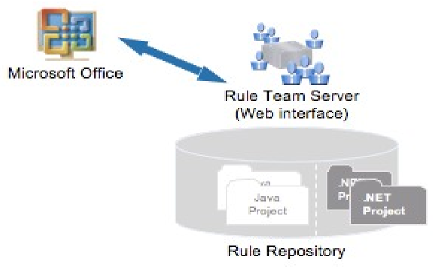
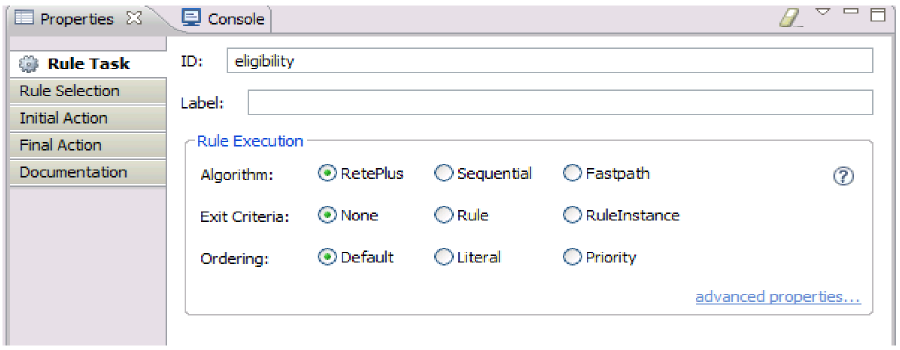


Fig.5.14: Rule Solutions for Office [9]

5.2.6 Rule Engine

A noticeable feature of JRules is that the modeler of the ruleflow can decide for each rule task, how the rule engine should execute it.



ID: eligibility

Label:

Rule Execution

Algorithm: [✓ RetePlus] Sequential Fastpath

Exit Criteria: [✓ None] Rule RuleInstance

Ordering: [✓ Default] Literal Priority

Fig. 5.15: Configuration options for the Rule Engine

<u>Algorithm:</u>

The user has the option of choosing one of three algorithms:

- **RetePlus** algorithm, which is basically the Rete Algorithm described in the previous chapter with some minor enhancements for JRules.
- **Sequential** algorithm
- **Fastpath** algorithm, which is a mixture of RetePlus and Sequential. It doesn't work with an agenda but still creates a network of nodes taking into account the similarity in the structure of the rules.

<u>Exit Criteria:</u>

This option let's the modeler define, when to stop the firing of the rules, which are in the agenda.

- **None**: The ordering scheme of the Rule Algorithm is applied. In the RetePlus mode all rules in the agenda are fired, until it is empty.
- **Rule**: Only the instances of the first applicable rule is fired, which is the one with the highest priority.

– **RuleInstance**: Only the first instance of the first applicable rule is fired.

Ordering:

This option let's the modeler define the way how applicable rules in the agenda are prioritized.

– **Default**: The ordering of the Rule Algorithm is applied.

– **Literal**: The order of the rules is taken from the order they are written down by the modeler.

– **Priority**: The modeler can give a priority to each rule in the rule task which let's the rule with the highest priority be fired first.

5.3 Integration of JRules and WPS

Since WPS is a runtime environment for an end-to-end business process and JRules can be part of this process representing the business policies, the two products are complementary.

To integrate JRules with the WebSphere Process Server the design tool WebSphere Integration Developer is used, since the processes which are developed in the WID will be deployed and executed on the Process Server.

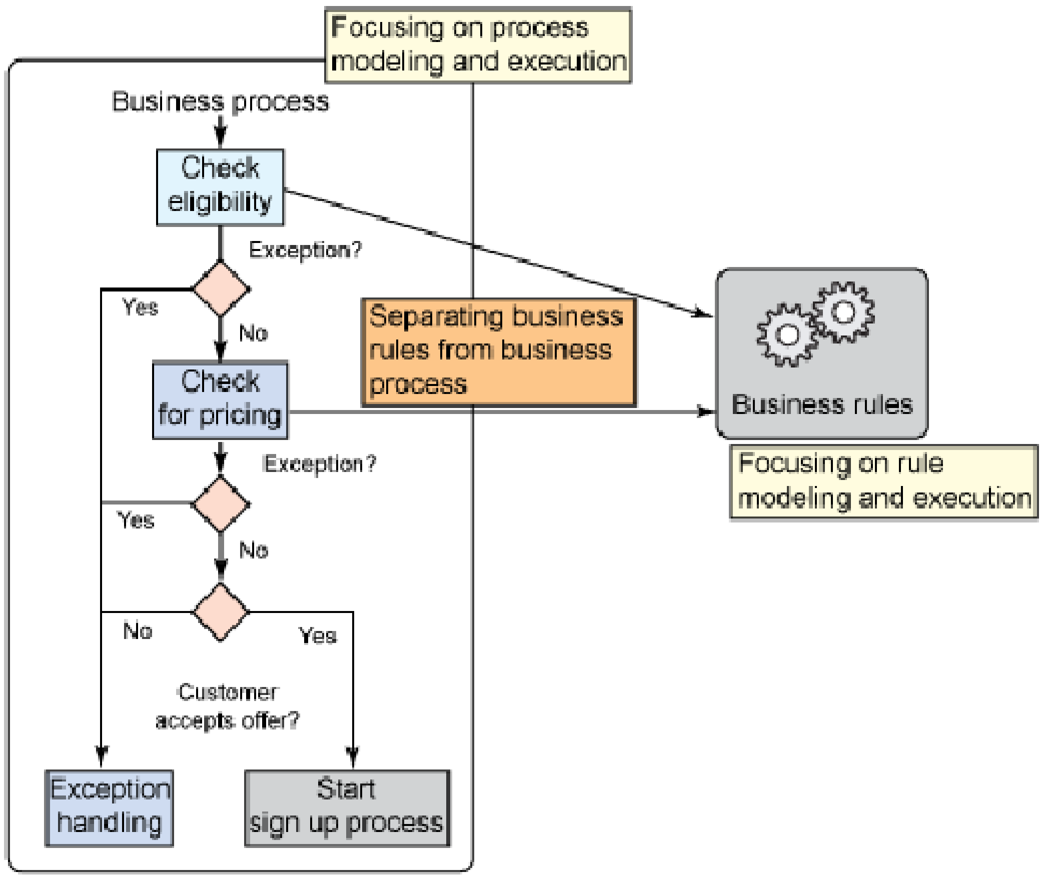


Fig. 5.16: Integration of JRules in a Business Process [21]

There are two ways for integrating JRules with WID. One is using the so called "Hosted Transparent Decision Service", the other one is using the "Decision Service Wizard", which are described in the following:

5.3.1 Hosted Transparent Decision Service:

The Hosted Transparent Decision Service (HTDS) makes a ruleset which has been modeled and tested in JRules available as a Web Service. Since rulesets are executed in the Rule Execution Server (RES) the Web Service provides the service endpoints within RES so that the service consumer can trigger the execution of the ruleset.

So HTDS is basically an enterprise application which provides a WSDL-file which describes how the service can be consumed(see the chapter about Web services for details). [Once the Service has been called, a session bean again calls the ruleset in the rule application which has been deployed to the Rule Execution Server.]

The WSDL-file is imported into the WID. The WID creates, conforming to the Service Component Architecture, the interface which can then be attached to the mediation module. Also the business objects used in the rule are created in WID.

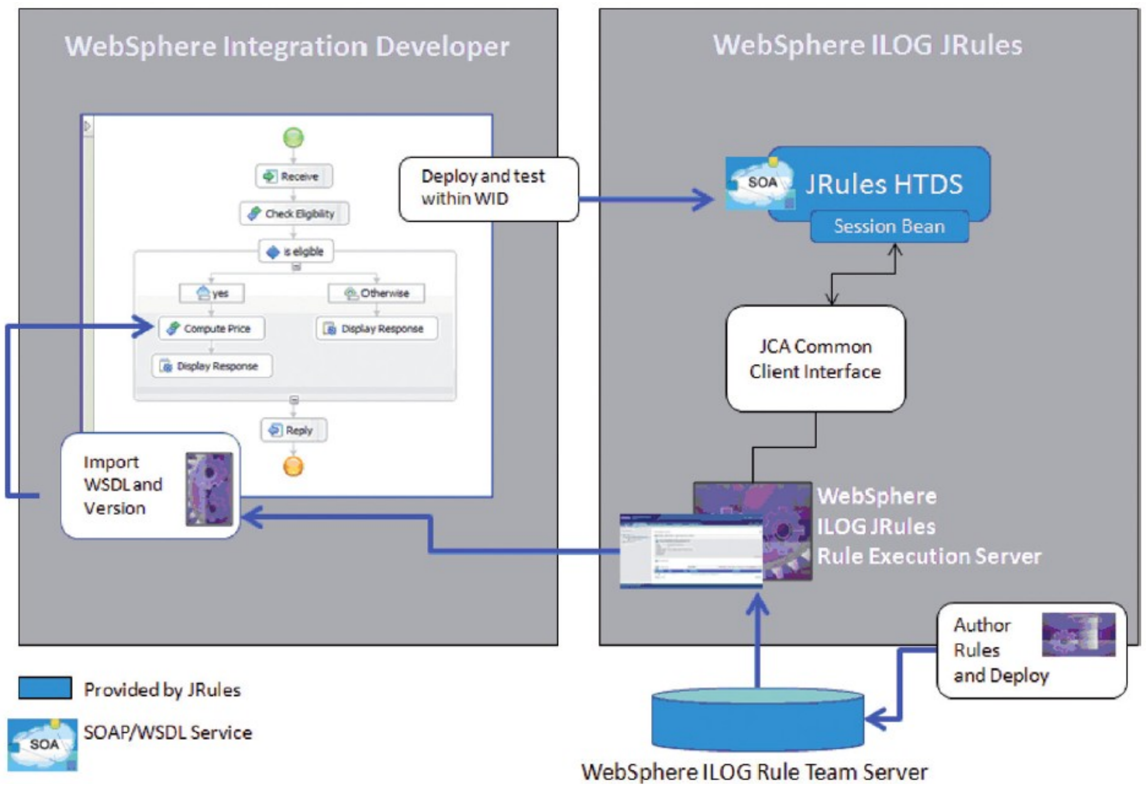


Fig. 5.17: Integration of JRules in WID as a Web Service [15]

In JRules it can be chosen between either to get the WSDL for a specific ruleset version or for the latest ruleset version.

5.3.2 Decision Service Wizard:

The other option to use JRules in WID is using the "Decision Service Wizard" Option.

As mentioned, in a Service Component Architecture a process is made up of components which are grouped together into deployable units – the modules.

With the "Decision Service Wizard" JRules will be used like a component, just like other components, like e.g. human tasks. It will have an interface where the functionality is offered through defined formats for input and output data. The component is called a "decision service".

In order to be able to use this wizard the JRules application has to be deployed on the WebSphere Application Server and it then has to be configured. The configuration comprises the set up of the connection to the data base where the rule apps are stored. A data source has to be configured in the application server, i.e. a connection pool, which handles the connections to the database. Also some user roles, like admin or business user, have to be set up in the WAS - and the JNDI names have to be configured.

And last but not least the Resource Adapter has to be installed on the WAS. The Resource Adapter makes the access to execution engine of JRules possible and handles the low level details of rule execution. It also makes runtime data available to the management console for the monitoring of the rules.

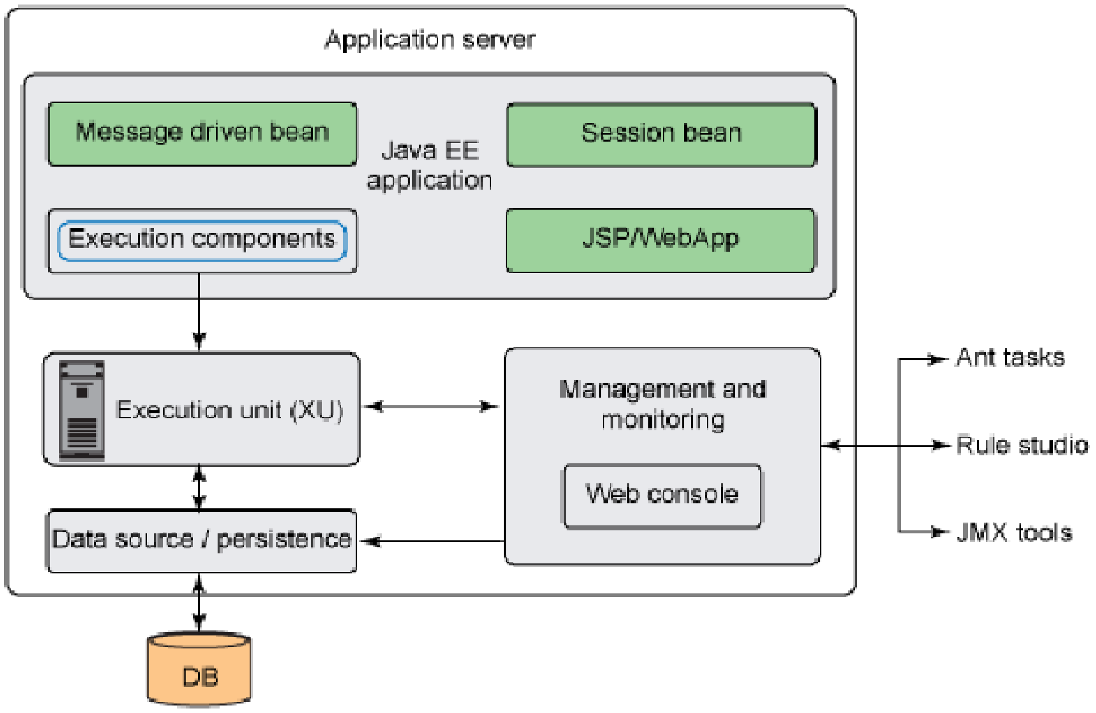


Fig. 5.18: Integration of the Rule Execution Server in WAS through a Resource Adapter (XU) [21]

The Resource Adapter of JRules is called “Execution Unit” (XU) and is called by an application through the “Execution Components”, which have to be embedded in the application through the import of a jar-file (Java Archive).

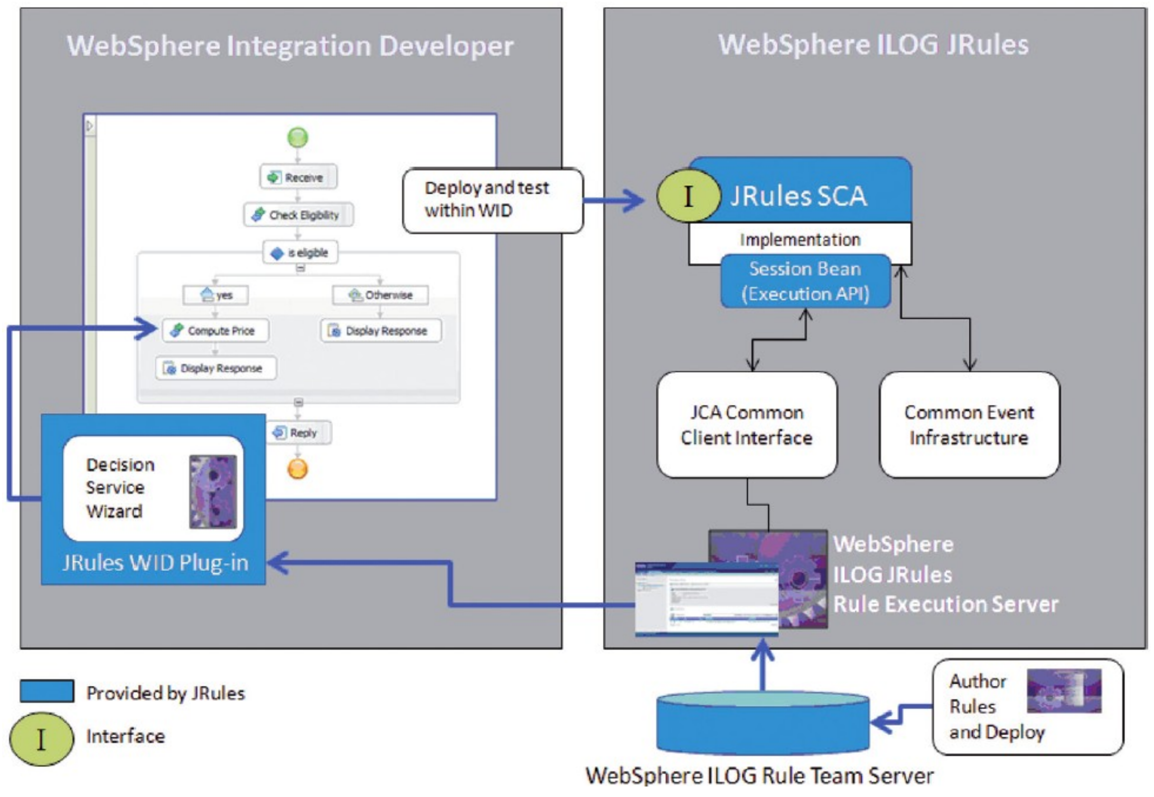


Fig. 5.19: Integration of JRules in WID as an SCA component [15]

5.3.3 Decision between HTDS and Decision Service Wizard

The question whether to use HTDS or the Decision Service Wizard to Integrate JRules with WPS depends on some aspects, which are summarized in the table below [22].

If the XOM is based on an XML-Schema and you want to expose and consume the rules as a web service which uses the XML-based SOAP protocol for information exchange, then using HTDS is the right choice. On the other hand the use of the Decision Service Wizard is necessary, if the XOM consists of Java Models. In this case it is not possible to expose the rulesets as web services.

Also if the rule engine is going to be called as a stateless session bean the Decision Service Wizard has to be used, because the rule engine is integrated there directly in the application server.

	HTDS	**Decision Service Wizard**
XSD Model	X	X
POJO Model		X
SOAP	X	
EJB		X

Table 5.1.: Decision between HTDS and Decision Service Wizard

6 SAP BRMS and BPM Solutions

Similar to the previous chapter,this chapter provides an overview of SAP's ecosystem, the Business Rule Management Systems SAP offers and how both are integrated.

Unlike IBM, SAP offers software based on two technologies, ABAP and Java. Therefore it has one application server based on ABAP and one based on Java, which is also the case with the BRMS. As mentioned in the second chapter, SAP has it's key business in selling enterprise software applications, mainly ERP software. This chapter starts with a historical overview of SAP's key products and how the principles of the Service Oriented Architecture are adopted.

6.1 SAP R/3

SAP's success in selling business applications started in 1993 with the offering of a distributed ERP Software Suite based on the Client-/Server-Architecture, called R/3. ERP is short for "Enterprise Resource Planning" and enables a coherent view on all applications which are relevant for a company's business. R/3 consisted of a set of complementary and reusable business applications which could be customized to meet special customer needs.

The core application was the Enterprise Resource Planning (ERP) software. Other optional applications were Supply Chain Management (SCM), Customer Relationship Management (CRM) and Human Resources (HR), which could be installed and run separately, but extracted data from the same database. While the previous version of SAP's ERP Software R/2 was running on mainframes, the Client-Server Architecture of R/3 made it compatible with Windows- and Unix-Systems, which opened up a whole new customer base.

All applications of SAP were programmed in a proprietary programming language called "ABAP" (Advanced Business Application Programming).

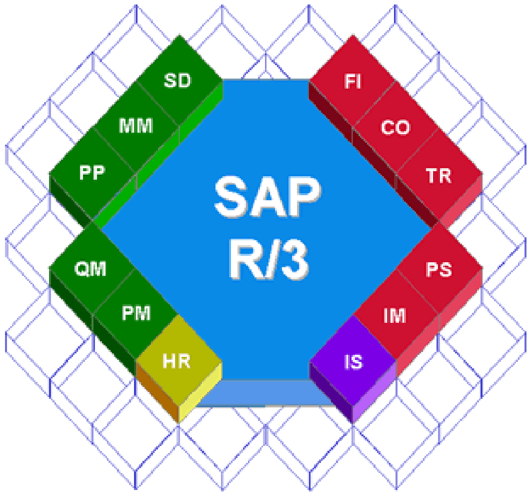


Fig.6.1: Modular Design of SAP R/3 [Source:IBM]

6.2 SAP Business Suite

With the introduction of the Service Oriented Architecture in the early 2000's SAP started to develop products for the middleware market through it's new SOA-platform called SAP NetWeaver. While customers who bought R/3 had to completely replace their existing R/2 software, the purpose of SAP NetWeaver was to keep the functionality of R/3 and integrate it with other applications in the customer's IT environment. This was achieved by exposing the R/3 modules as enterprise services.

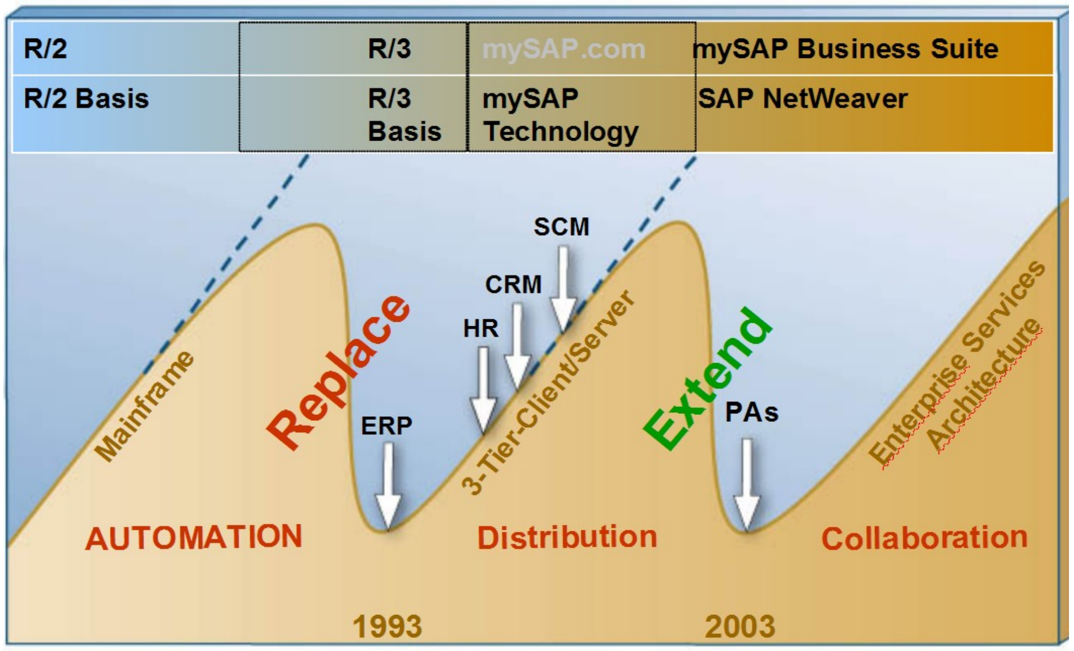


Fig. 6.2: Existing investments in R/3 are saved when adopting SAP NetWeaver* [Source:IBM]

* PA= Packaged Application Component

The purpose of SAP NetWeaver is to offer an SOA-based platform for the integration of existing enterprise applications and flexible integration of new applications, independent of the platform they have been built on. This results in a lower TCO, which sets free new budgets for innovations.

With the introduction of SAP NetWeaver SAP R/3 was released as "mySAP ERP" and the integrated product of mySAP ERP and SAP NetWeaver was called "mySAP Business Suite", which should enable customers to model business process, which could span over all applications in the IT landscape.

SAP NetWeaver tackles these demands of holistic integration by offering a platform which unifies different systems where the systems make use of the same data (SAP Master Data Management), offer the same user interface (SAP Enterprise Portal) and the same process (SAP Process Integration), amongst others. Additionally the performance of the consolidated process can be measured using SAP Business Intelligence (BI).

All the Java based components of SAP NetWeaver are deployed on the J2EE certified Application Server of SAP, while the ABAP based components are deployed on the ABAP Application Server, which are both available in NetWeaver.

As explained in Chapter 3, in an SOA the applications in a company are exposed as services and a business process is built through the orchestration of these services, which is often executed using the WS-BPEL standard.

SAP now has it's own definition of an SOA, which is called "Enterprise SOA" (ESOA) and differs a bit from the standard SOA:

In an ESOA the services which are orchestrated within a process do not offer the single functionality of an application, instead they offer the composed functionality of multiple services which together fulfill a certain task. For example if an order in the CRM application has to be canceled then consequently the respective order has to be canceled in the SCM application. These two related tasks are distinct services, which together build a micro-process. Because of the fact, that the independent exposure of these tasks as distinct services is not reasonable, SAP exposes the whole micro-process as a single service and calls this composed service an "Enterprise Service", whereas distinct services, which prove a single functionality, are called "Application Services". This method to expose services is part of SAP's interpretation of a SOA, called ESOA (Enterprise Service Oriented Architecture).

ESOA has the following 5 key principles [22]:

– **Abstraction**: hiding technical details that can be confusing

– **Modularity**: breaking down complexity and creation of reusable pieces

– **Standardized connectivity**: enabling flexible composition of services to form processes

– **Loose coupling**: allowing for separate evolution of the various components without breaking any points of integration

– **Incremental design**: enabling changes to composition and configuration without affecting the internals of components, and vice versa

Because in ESOA processes are built through the orchestration of Enterprise Services the Services are consequently registered in an "Enterprise Repository" where they can be looked up for consumption.

6.3 SAP NetWeaver BRM

SAP NetWeaver Business Rules Management is SAP's Java-based BRMS, which originates in the acquisition of Indian company "Yasu Technologies" in 2006, which offered a BRMS called "QuickRules". SAP NetWeaver offers a full rule lifecycle management which can be used to automate decisions either in business processes or applications. It is shipped as part of SAP's BPM tool called "SAP NetWeaver Composition Environment", which is described in the following (version 7.2).

6.3.1 SAP NetWeaver Composition Environment

As mentioned, in SAP NetWeaver the modules of the SAP Business Suite are exposed as enterprise services. They are called "Core Processes", because they run the core business and have the characteristic, that their stability and efficiency is more important than their ability to respond to continual change. But the customer's business processes normally consist of more applications than the Core Processes and the purpose of ESOA is to integrate them together with the Business Suite. This integration of SAP's core processes with non-SAP, legacy or B2B processes is handled by "SAP Process Integration" (SAP PI), which is also a part of SAP NetWeaver.

Besides the "Core Processes" SAP wants to provide a tool for the creation of the so called "Edge Processes" which are applications that change more often, like for example Business Rules. This tool is called "SAP NetWeaver Composition Environment" (SAP NW CE) and enables the modeling of Business Processes, which contain both Edge and Core

Processes, thus making it a BPM tool. As distinct from SAP PI, SAP CE is optimized for human-centric processes, therefore emphasizing the flow of human tasks.

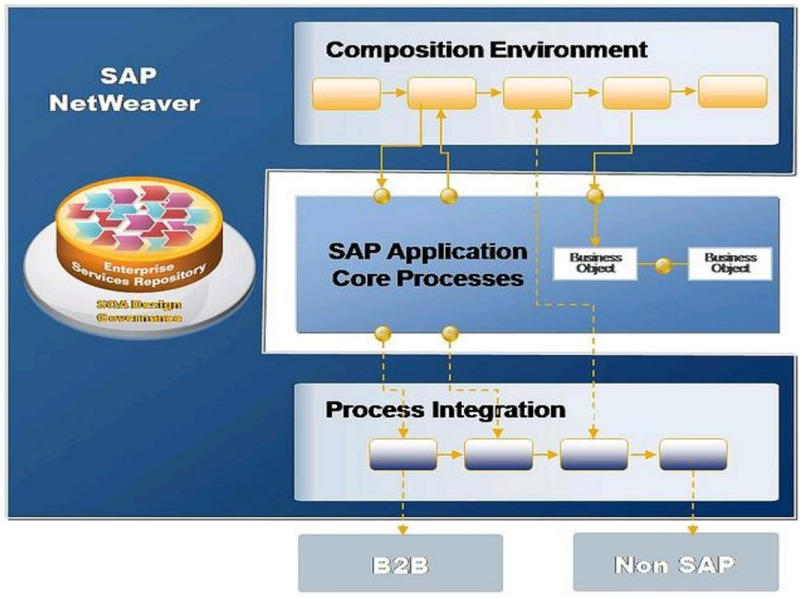


Fig. 6.3.: SAP NetWeaver CE is a BPM tool connecting SAP's Core and Edge Processes

SAP's Component Model

In SAP NetWeaver CE applications are developed using SAP's Component model, which is defined as follows:

> "SAP's component model is a programming model for software. The idea is that by dividing software projects into different components, you can organize them right from the start into comprehensible and reusable units. Components can use other components in a well-defined and controlled manner, encapsulate child components, or publish their functions in a set of public interfaces (the so called public parts). " (Source:SAP)

The main components a software consists of are Software Components (SC) and each Software Component is again made up of several Development Components (DC), as illustrated in the picture.

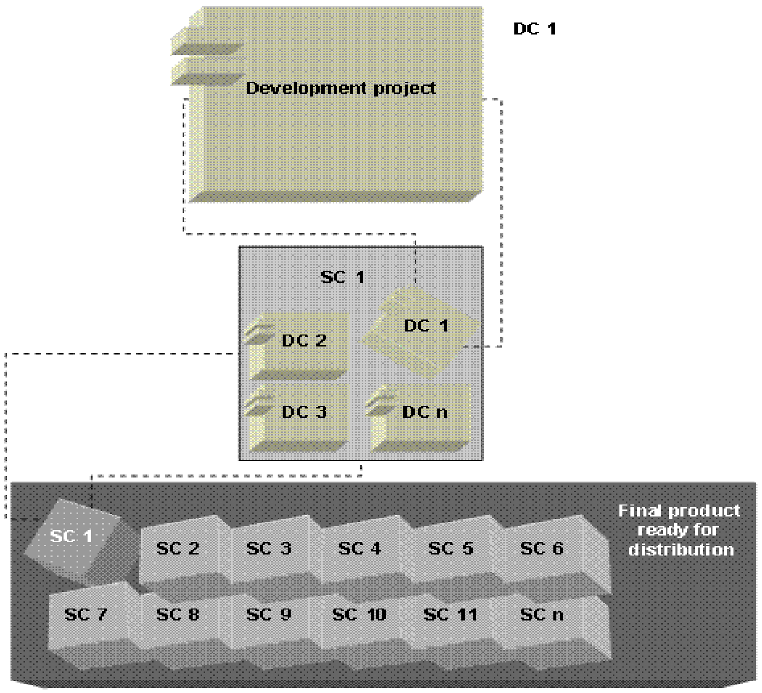


Fig. 6.4: Design of SAP's Component Model [23]

A Development Component is technically the same as a Java Project. There are many types of DC's depending on which part of the SC is being worked on. For example EJBs are implemented using the EJB-DC and plain Java Projects are implemented using the Java-DC.

Additionally a DC can share some parts with other DC's in the same SC, for example a number of Java classes in a Java-DC. This is done by exposing these parts as "public parts" which are interfaces provided to the other DC's. The other DC's who want to make use of this interface define a "dependency" to the DC.

6.3.2 Environments

Conforming to SAP's Component Model the rules are developed in a Development Component (DC) called "Rules Composer".

There are two environments for authoring and managing business rules, namely the “Rules Composer” and the “Rule Manager”. For saving and sharing rules between different applications the modeled rules are saved in a repository.

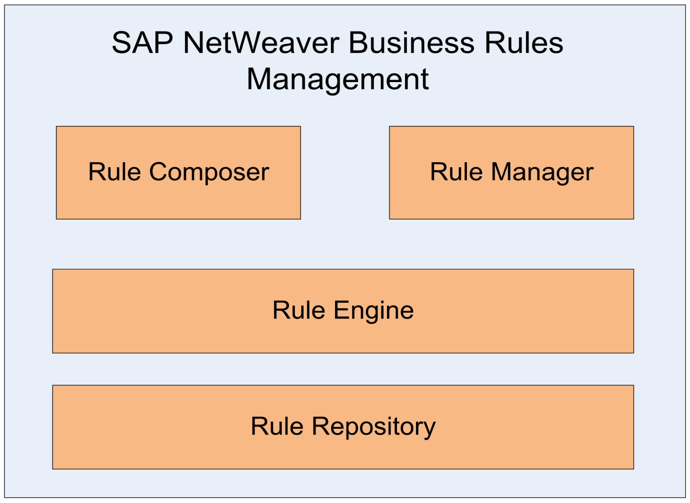


Fig. 6.5: Components of SAP NW BRM

Rules Composer:

The Rules Composer is an Eclipse-based tool providing IT Developers an environment for creating business rules. It is integrated in the SAP NetWeaver Studio as a separate perspective.

Rules can be created in two ways:

– In the Process Composer: This couples the business rules to the lifecycle of the Business Process modeled in the Process Composer. This helps in centralizing the management of the process with it's rules but prevent the reusablity of the rules in other processes.

– In the Rules Composer: Here the rules are modeled in a separate perspective, which enables the usage of rules by any other business application or business process.

Rules Manager:

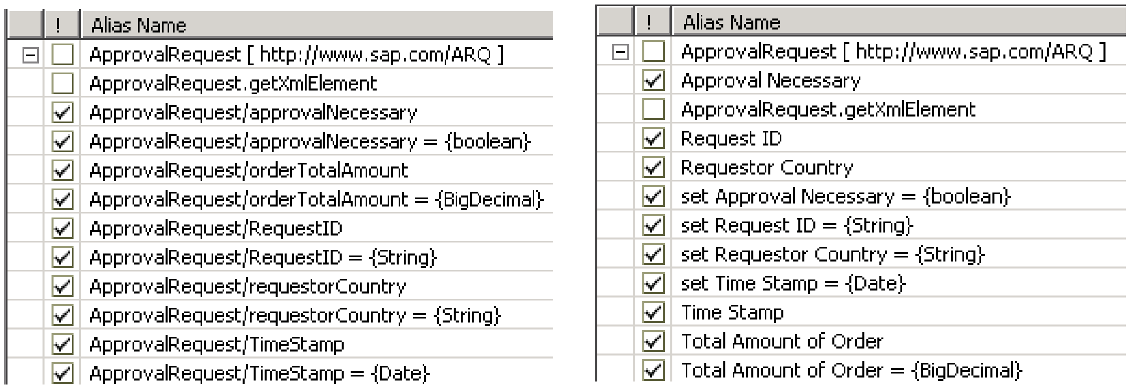
The Rules Manager provides a user interface based on SAP's Web Dynpro technology for business users to manage and update the rules, which have been created in the Rules Composer.

6.3.3 Verbalization

Like any other BRMS the objects have to be defined which the Rule Engine will work on.

These can be either defined in an XML Schema or in Java Classes and have to be imported to the Rule Project. Naturally these objects are not directly suitable for modeling business rules, because they are not readable. Therefore a mapping should be done to custom expressions, defined by the user. In SAP NW BRM this step is not mandatory, but highly recommended.

Below is an example of the mapping from objects defined in an XML-Schema to custom expressions. It can be seen that the readability is limited due to the nested structure of an XML-File.



	!	Alias Name
⊟	☐	ApprovalRequest [http://www.sap.com/ARQ]
	☐	ApprovalRequest.getXmlElement
	✓	ApprovalRequest/approvalNecessary
	✓	ApprovalRequest/approvalNecessary = {boolean}
	✓	ApprovalRequest/orderTotalAmount
	✓	ApprovalRequest/orderTotalAmount = {BigDecimal}
	✓	ApprovalRequest/RequestID
	✓	ApprovalRequest/RequestID = {String}
	✓	ApprovalRequest/requestorCountry
	✓	ApprovalRequest/requestorCountry = {String}
	✓	ApprovalRequest/TimeStamp
	✓	ApprovalRequest/TimeStamp = {Date}

	!	Alias Name
⊟	☐	ApprovalRequest [http://www.sap.com/ARQ]
	✓	Approval Necessary
	☐	ApprovalRequest.getXmlElement
	✓	Request ID
	✓	Requestor Country
	✓	set Approval Necessary = {boolean}
	✓	set Request ID = {String}
	✓	set Requestor Country = {String}
	✓	set Time Stamp = {Date}
	✓	Time Stamp
	✓	Total Amount of Order
	✓	Total Amount of Order = {BigDecimal}

Fig. 6.6: Objects from an XML Schema before verbalization (left) and after verbalization

The Schema is imported to the Rule Project and the verbalization is done in the “Aliases” section. From then on the verbalized objects are ready to be used.

Importing objects from Java Classes is different. The Java Classes which should be used in the Rule DC reside in a Java DC, therefore a dependency has to be created between both DCs. This is done by adding the desired classes to the public part of the development component in order to be able to expose them to the Rule-DC.

Once this is done the DC of the Rules Project has to add a dependency to the DC where the Java classes have been exposed and the Rule DC has to be built afterwards.

Finally the classes can be added in the Alias section and be verbalized. The verbalization is done through the getter- and setter-methods of the classes. If a variable of the class does not have a getter or setter then it can't be verbalized.

6.3.4 Modeling

Except Decision Trees, the basic constructs for modeling rules are provided by SAP NW BRM.

Business Rules:

The creation of a business rule is done by simply defining a condition and an action, like the following example:

```
Rule : ApprovalNeededRule
Priority : 50000
Overrides :
Effectivity : Always
<Click to enter comments>
Preconditions :
+
If
Boolean.TRUE Equals Boolean.TRUE
+
Then
Evaluate-DecisionTable :: ApprovalNeededDecisionTable
```

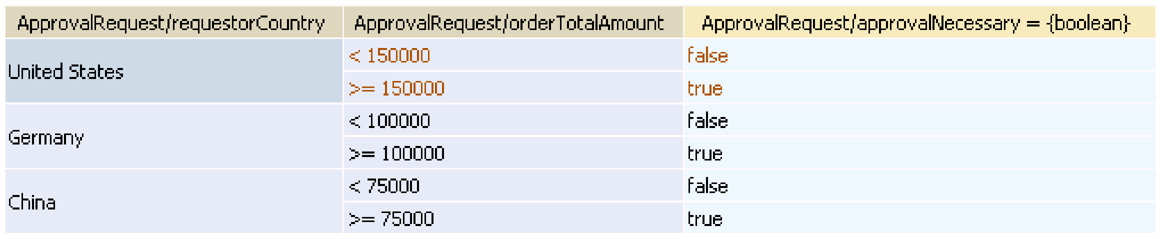
As can be seen it is possible to set a priority for each business rule, so that the rule engine can decide which rule to fire first, if the conditions of more than one rule is satisfied. Through the “Effectivity”-option it can be chosen whether to deactivate a rule, so that it is never fired, or to make it always available for execution. Also preconditions can be set, like defining constant variables for the rule creation.

In the example above the condition of the rule is always met, therefore the Action-Part of the Rule is always evaluated which, in this case, evaluates a decision Table.

<u>Decision Tables:</u>

Decision Tables are created using a Wizard, where the columns of the decision table are defined. There are two types of columns, condition columns and action columns.

If a row, which can span multiple condition columns, applies, then the corresponding row in the action column is executed.



ApprovalRequest/requestorCountry	ApprovalRequest/orderTotalAmount	ApprovalRequest/approvalNecessary = {boolean}
United States	< 150000	false
	>= 150000	true
Germany	< 100000	false
	>= 100000	true
China	< 75000	false
	>= 75000	true

Fig. 6.7: A Decision Table in SAP NW BRM

The terms which are used for the column headers are taken from the user defined expressions in the "Aliases". If no Aliases have been defined, then the columns have unreadable names, like in the example above.

Decision Tables can't be evaluated directly, therefore they have to be combined with a business rule.

<u>Flow Rules:</u>

In a Flow Rule an order can be defined in which the modeled rules will be executed, using a graphical tool. The type of rules which can be executed in a Flow Rule are simple business rules, decision tables, rule scripts (see below) and other flow rules, which enables a nested hierarchy.

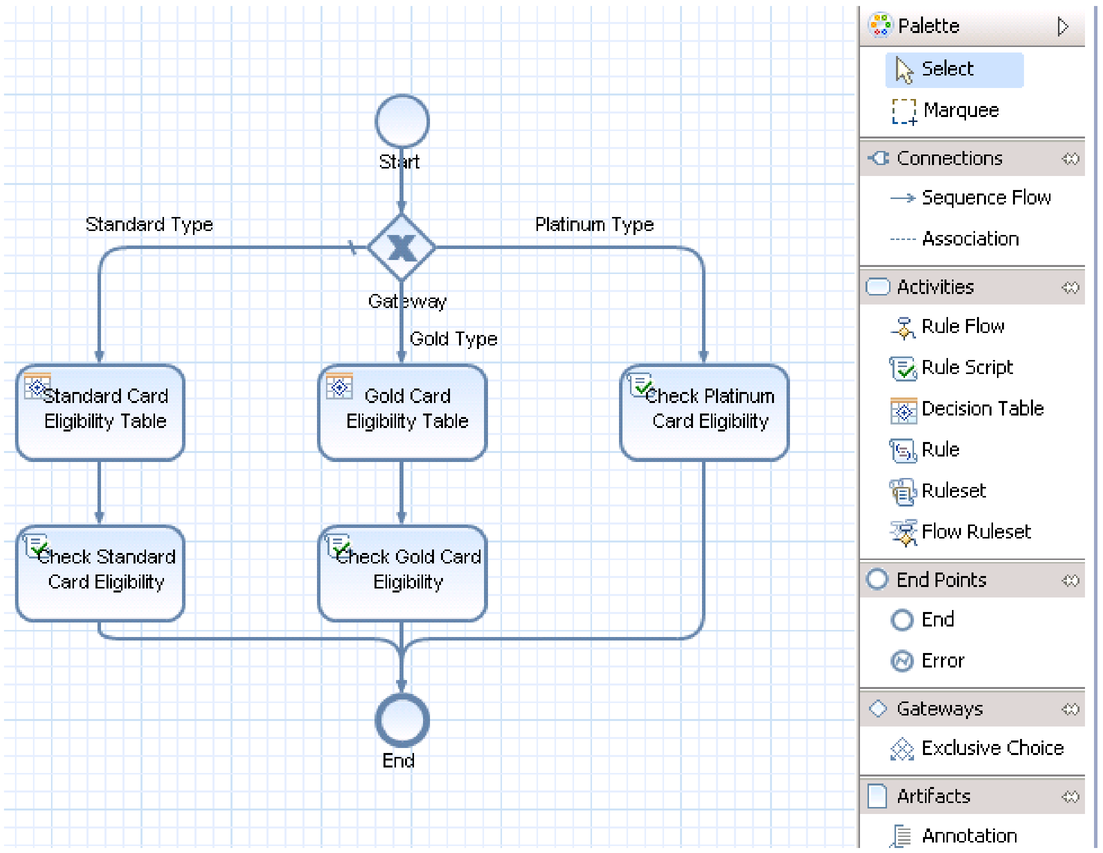


Fig. 6.8.: Example of a flow rule which checks the eligibility of a customer for a certain requested credit card (Standard Card, Gold Card or Platinum Card)[24]

Rule Scripts:

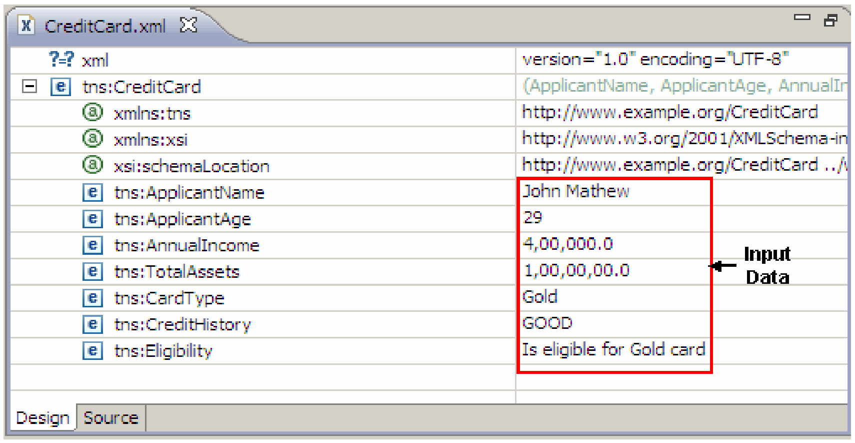
Rule Scripts define a series of actions which are executed when the preceding steps in the rule flow are satisfied. They can also include loops, for example a while-loop, which allows to use technical constructs for the rules.

6.3.5 Testing

In SAP NW BRM there is the possibility to test the modeled rulesets either by running a "Test Scenario" which contains one set of simulated data or a "Test Case" which contains multiple sets of simulated data.

Test Scenario:

In a Test Scenario the input data for the scenario is typed directly next to the Objects in the Schema File and then executed (see screenshot).



?-? xml	version="1.0" encoding="UTF-8"
tns:CreditCard	(ApplicantName, ApplicantAge, AnnualIr
xmlns:tns	http://www.example.org/CreditCard
xmlns:xsi	http://www.w3.org/2001/XMLSchema-in
xsi:schemaLocation	http://www.example.org/CreditCard ../
tns:ApplicantName	John Mathew
tns:ApplicantAge	29
tns:AnnualIncome	4,00,000.0
tns:TotalAssets	1,00,00,00.0
tns:CardType	Gold
tns:CreditHistory	GOOD
tns:Eligibility	Is eligible for Gold card

Fig. 6.9: A Test Scenario [24]

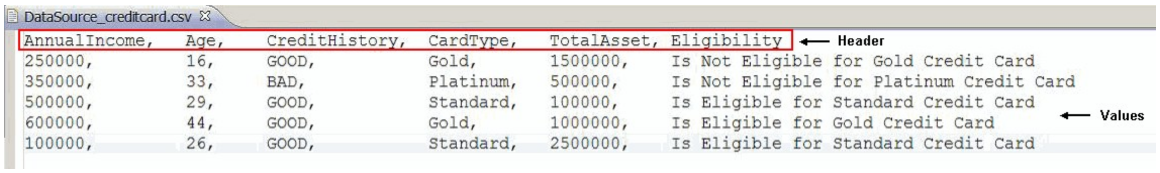
Test Case:

A test case is composed of:

– **Test target**: The test target is the ruleset which contains the rules that need to be tested.

– **Test data**: The test data is written in a CSV-File (Comma-Separated Values) and contains a table with column headers and a number of rows. Each row represents a test-case with input data and expected output data.

– **Test mapping**: The column headers in the CSV-File contain custom expressions which make the test-file more readable. Therefore they have to be mapped to the Aliases which are used in the Rule Composer using Xpath.



```
AnnualIncome,   Age,   CreditHistory,  CardType,   TotalAsset,  Eligibility
250000,         16,    GOOD,           Gold,       1500000,     Is Not Eligible for Gold Credit Card
350000,         33,    BAD,            Platinum,   500000,      Is Not Eligible for Platinum Credit Card
500000,         29,    GOOD,           Standard,   100000,      Is Eligible for Standard Credit Card
600000,         44,    GOOD,           Gold,       1000000,     Is Eligible for Gold Credit Card
100000,         26,    GOOD,           Standard,   2500000,     Is Eligible for Standard Credit Card
```

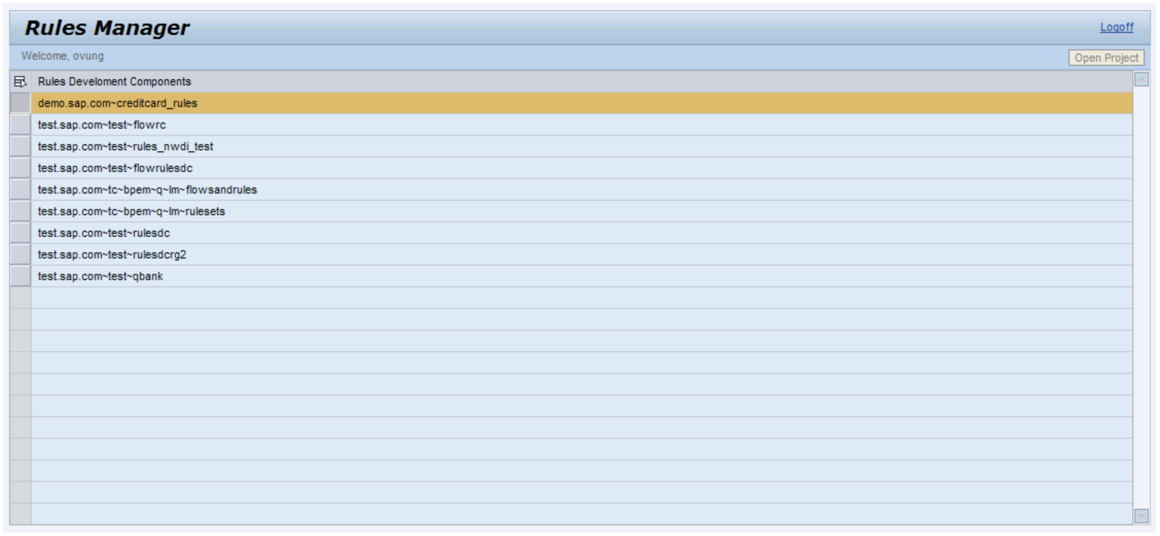
Fig. 6.10: A Test Case written in a CSV-File [24]

6.3.6 Rules Management for Business Users

As mentioned, the environment for business users to manage the rules created in the Rules Composer is the Rules Manager.

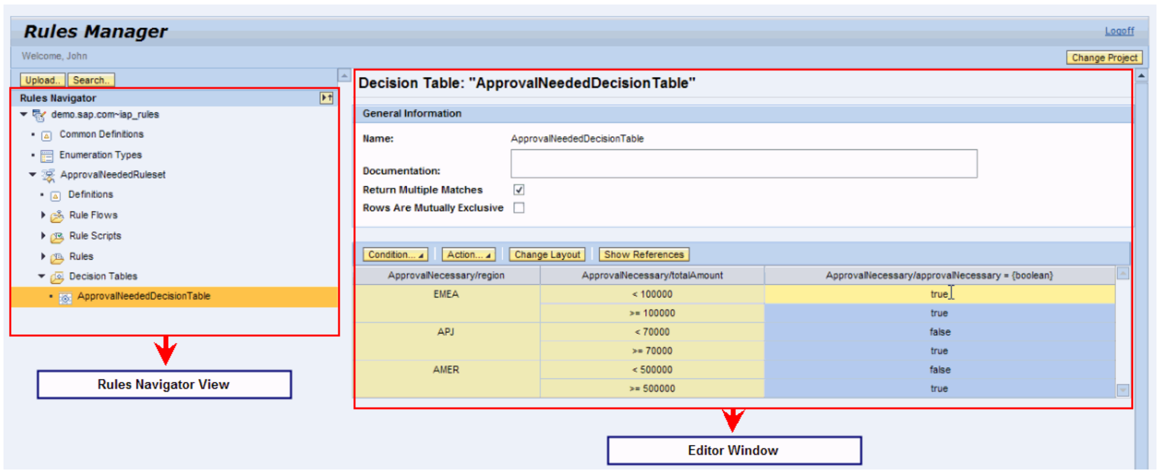
The main features of the Rules Manager are:

- Access Control: This feature enables a role-based user access to the rule projects. If a user is not explicitly authorized for certain rule projects, they can't be read or modified by this user.
- Editing Tool for Business Users: The Rules Manager enables authorized business users and administrators to modify business rules and update them to the production system
- Version History: Different versions of Rules can be compared by checking the version history of the rules.
- Report Generation: Reports can be generated of rulesets, rules and decision tables.
- Rules Deployment: Even as the application is running, rules can be changed and deployed by business users in real time



Rules Develoment Components
demo.sap.com~creditcard_rules
test.sap.com~test~flowrc
test.sap.com~test~rules_nwdi_test
test.sap.com~test~flowrulesdc
test.sap.com~tc~bpem~q~lm~flowsandrules
test.sap.com~tc~bpem~q~lm~rulesets
test.sap.com~test~rulesdc
test.sap.com~test~rulesdcrg2
test.sap.com~test~qbank

Fig. 6.11: A number of projects in the Rules Manager



Decision Table: "ApprovalNeededDecisionTable"

Name: ApprovalNeededDecisionTable

Documentation:

Return Multiple Matches ✓

Rows Are Mutually Exclusive ☐

ApprovalNecessary/region	ApprovalNecessary/totalAmount	ApprovalNecessary/approvalNecessary = {boolean}
EMEA	< 100000	true
	>= 100000	true
APJ	< 70000	false
	>= 70000	true
AMER	< 500000	false
	>= 500000	true

Fig. 6.12: Environment for Business Users to manage the modeled rules [24]

Integration with Microsoft Excel:

SAP NW BRM offers the functionality to export modeled decision tables as Excel-Sheets, which can then be authored, edited and analyzed with Microsoft Excel. Other types of rules than decision tables can't be exported, like for example if-then rules for Microsoft Word.

6.3.7 Rule Engine

As an algorithm for Pattern Matching the Rule Engine of SAP NetWeaver BRM only offers the Rete algorithm for the execution of a ruleset.
Considering the execution order in the agenda, for each rule a priority can be assigned, so that the rules with a higher priority will be executed first. For the case, that two rules with the same priority are eligible to be fired, the "`override`" option is provided for each rule, which makes the engine fire this rule and ignore the other one.

6.4 BRFplus

BRFplus is short for Business Rules Framework plus and represents the BRMS for applications written in ABAP (Here we use the version shipped with SAP NW 7.0 EHP1).

As mentioned, ABAP stands for "Advanced Business Application Programming" and is the language which the Products of the SAP Business Suite are programmed in. The benefits that come with a BRMS are not bound to a certain programming language. And because most other BRMS are essentially built to support Java applications, it is logical that SAP has created a BRMS to support ABAP applications.

BRFplus is derived from BRF which was a tool for designing rules for the claims processing in insurance applications. At the beginning of its development BRFplus was known as the "Formula & Derivation Tool", that's why some artifacts in this tool contain the letters "FDT".

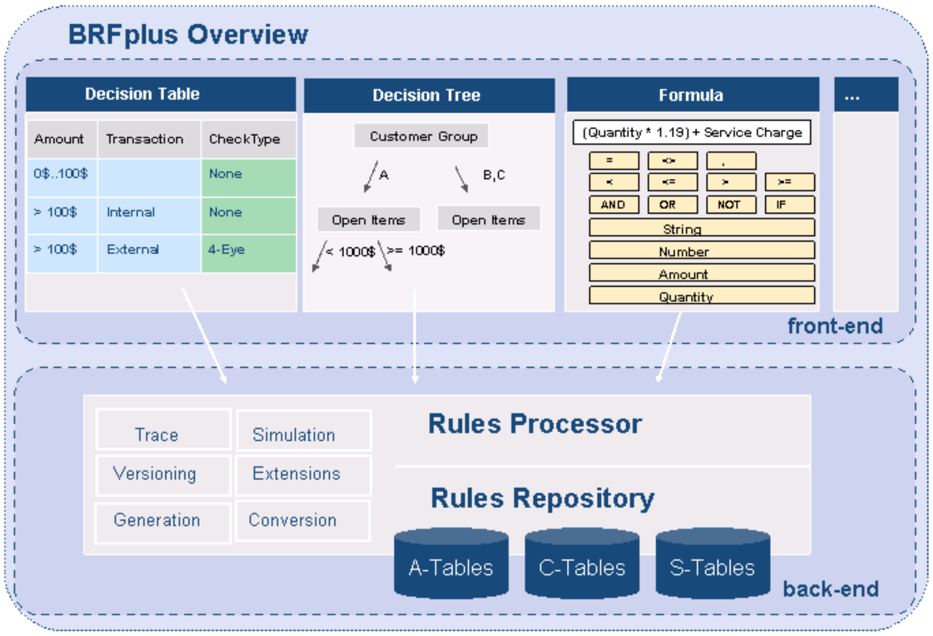


Fig. 6.13: BRFplus Overview. The Rule Engine is here known as Rules Processor[25]

6.4.1 Environments

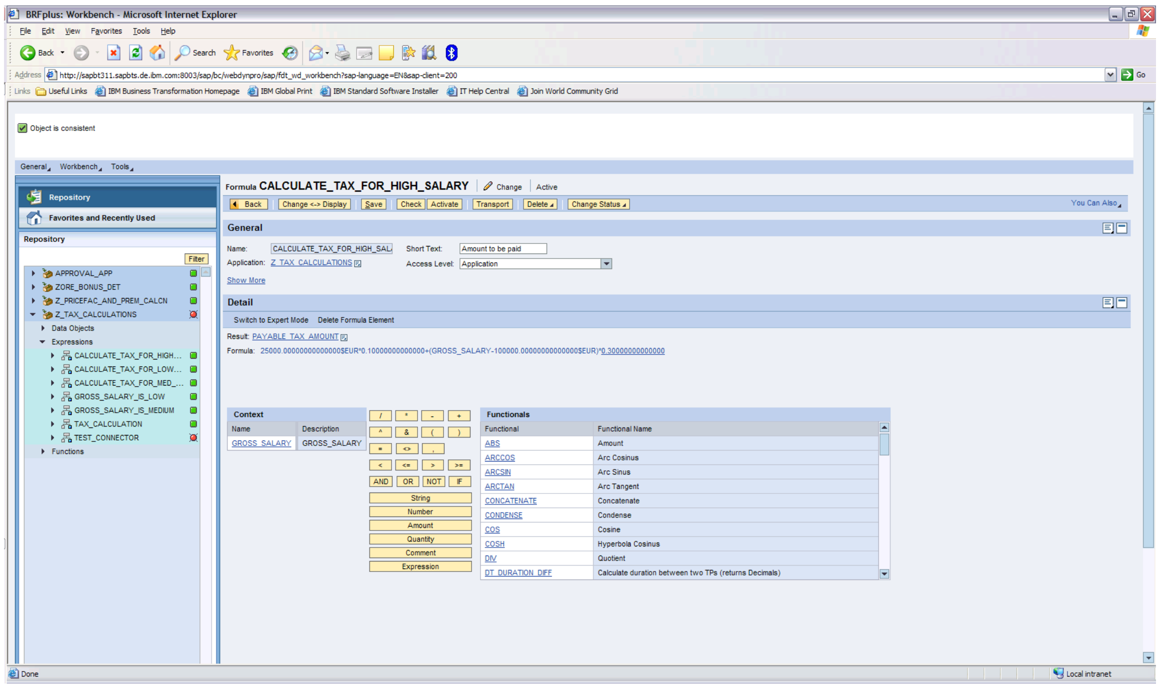
BRFplus provides only one user interface to model rules. This user interface is implemented using SAP's so called "WebDynpro"-Technology and is displayed in a Web Browser.

There are two modes of rules management:

– **Workbench mode:** This is the mode for IT developers in which the functionality of every rule in BRFplus can be seen.

– **Catalog mode:** This is the mode for Business Users. Here the Repository, which is the place where the objects and rules are stored, is not completely visible.

Only certain rules for authorized users are displayed and can be modified by them.


BRFplus: Workbench - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://sapbt311.sapbts.de.ibm.com:8003/sap/bc/webdynpro/sap/fdt_wd_workbench?sap-language=EN&sap-client=200
Links Useful Links IBM Business Transformation Homepage IBM Global Print IBM Standard Software Installer IT Help Central Join World Community Grid
Object is consistent
General Workbench Tools
Repository
Favorites and Recently Used
Repository
Filter
APPROVAL_APP
ZORE_BONUS_DET
Z_PRICEFAC_AND_PREM_CALCN
Z_TAX_CALCULATIONS
Data Objects
Expressions
CALCULATE_TAX_FOR_HIGH...
CALCULATE_TAX_FOR_LOW...
CALCULATE_TAX_FOR_MED_...
GROSS_SALARY_IS_LOW
GROSS_SALARY_IS_MEDIUM
TAX_CALCULATION
TEST_CONNECTOR
Functions
Back Change <-> Display Save Check Activate Transport Delete Change Status
You Can Also
Switch to Expert Mode Delete Formula Element
/ * - + ^ & ( ) = <> , < <= > >= AND OR NOT IF
String Number Amount Quantity Comment Expression
Done Local intranet

Formula **CALCULATE_TAX_FOR_HIGH_SALARY** Change Active

General

Name: CALCULATE_TAX_FOR_HIGH_SAL Short Text: Amount to be paid
Application: Z_TAX_CALCULATIONS Access Level: Application
Show More

Detail

Result: PAYABLE_TAX_AMOUNT
Formula: 25000.00000000000000$EUR*0.10000000000000+(GROSS_SALARY-100000.00000000000000$EUR)*0.30000000000000

Context

Name	Description
GROSS_SALARY	GROSS_SALARY

Functionals

Functional	Functional Name
ABS	Amount
ARCCOS	Arc Cosinus
ARCSIN	Arc Sinus
ARCTAN	Arc Tangent
CONCATENATE	Concatenate
CONDENSE	Condense
COS	Cosine
COSH	Hyperbola Cosinus
DIV	Quotient
DT_DURATION_DIFF	Calculate duration between two TPs (returns Decimals)

Fig. 6.14: Environment of BRFplus for Rules Modeling

6.4.2 Modeling

In BRFplus there are three modes of operation to execute a function:

– Functional mode

– Event mode

– Functional and Event mode

Functional mode:

In the functional mode a top expression is assigned to the function, which is evaluated first, whenever the function is called. An expression can have many forms, like a decision table, a formula expression or a case expression. Event a constant is an expression, it's evaluation will return a fixed value.

So in functional mode the top expression is evaluated and directly returns a result. This mode is normally used for simple use cases where the rules are not complex.

Event mode:

In event mode no expression is directly evaluated. Instead an event is triggered which can be subscribed by a number of rulesets. These rulesets will then perform some actions, for example modifying variables or writing messages to a log.

If multiple rulesets have subscribed to an event and this event is triggered, then all rulesets will be executed "in parallel". There is no functionality given in BRFplus to define an order in which the rules are executed.

Functional and Event mode:

This mode is a mixture of the functional mode and the event mode. It requires a top expression which does not have to return a value directly but can also trigger events and therefore rulesets. Some values can be modified and calculated which will be ultimately returned by the top expression as a result. So this mode can be seen as the function mode in a nested structure.

How BRFplus is called:

ABAP-Applications call BRFplus through six lines of code. These six lines contain the context data which serve as input to BRFplus and the variable in which the result of the decision is stored (see Appendix).

BRFplus also provides the creation of a web service which provides the functionality of a function also to non-ABAP applications or processes.

Creating Rules in BRFplus:

For the explanation of how to create rules the functional mode is used.

In BRFplus there are three types of Objects.

- Data Objects
- Expression Types
- Functions

Data Object:

A data object is the data the rule engine of BRFplus will work on. It includes the context data which is given by the calling application and custom defined data objects.

The data types which can be associated with a data object are:

- Amount
- Boolean
- Number
- Timepoint
- Quantity
- Text

Expression Type:

Expression Types represent the calculation power of BRFplus. They perform some calculation and return a result. Some built-in expression types are the following:

- Boolean
- BRMS Connector
- Case
- Constant
- Database Look-up
- Decision Table
- Formula
- IF-THEN Rules
- Random Number
- Static ABAP Method
- Value Range

BRFplus also provides the ability to create own Expression Types.

Function:

A function is the entity which is called by the application and which uses the expressions and data objects to deliver a result as output. As mentioned, functions can be executed in functional mode, event mode or a mixture of both.

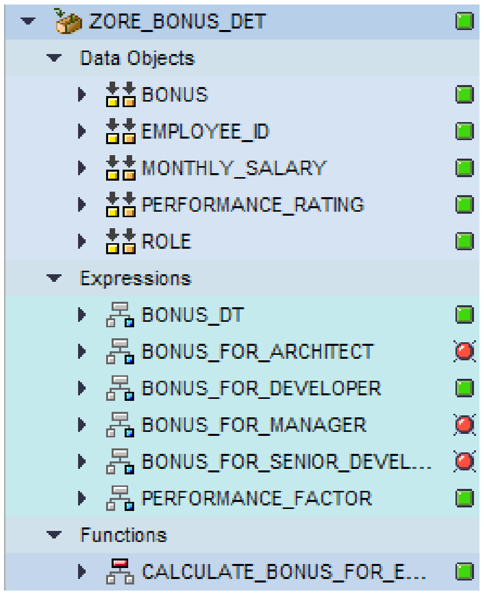


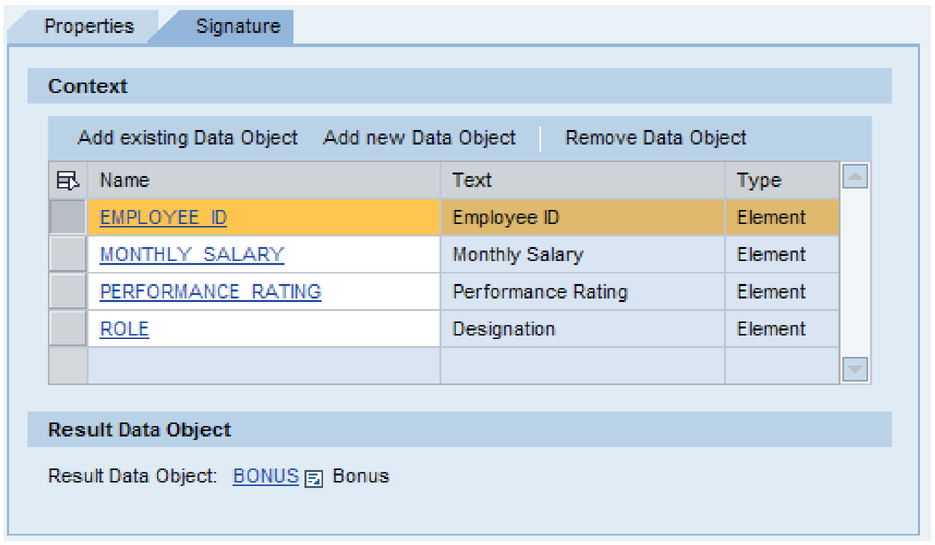
Fig. 6.15: Data Objects, Expressions and Functions in BRFplus with activation status

Before execution, Data Objects, Expressions and Functions have to be activated, which are then represented by green squares (see picture above).

The procedure of rule creation in BRFplus is the following:

- Creation of a function
- Definition of the execution mode for the function (Functional mode, Event mode, or both)
- Creation of data objects or import of existing objects
- Definition of the signature of the function by assigning the data objects
- Assignment of a data object as the result object of the function

– Assignment of a top expression to the function, which will be processed instantly whenever the function is called



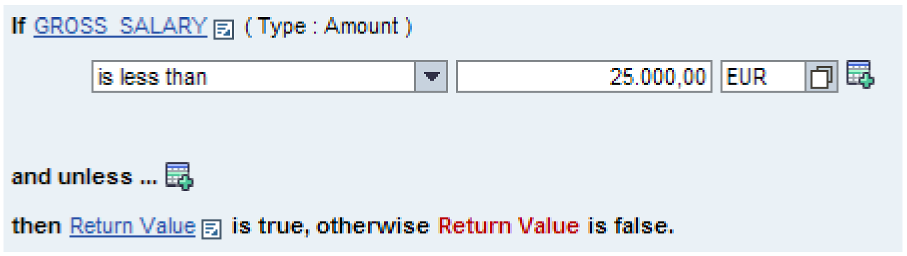
Name	Text	Type
EMPLOYEE_ID	Employee ID	Element
MONTHLY_SALARY	Monthly Salary	Element
PERFORMANCE_RATING	Performance Rating	Element
ROLE	Designation	Element

Result Data Object: BONUS Bonus

Fig. 6.16: Signature of a function

Business Rule:

Using the defined data objects, business rules can be modeled using the known "if-then-else" pattern. Additionally, it is possible to exclude conditions using the respective button shown in the screenshot.



If GROSS_SALARY (Type : Amount)

is less than 25.000,00 EUR

and unless ...

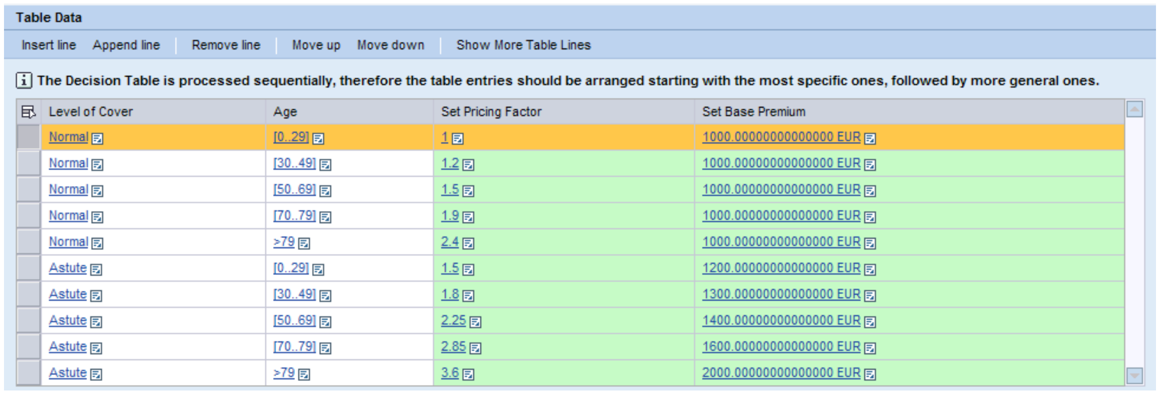
then Return Value is true, otherwise Return Value is false.

Fig. 6.17: A business rule in BRFplus

Decision table:

Data objects can be given an alternative name, which makes it more readable. One of the reasons is that the name of data objects always contain upper case letters.

These alternative names are then assigned to the column headers of the decision table. The decision tables contain two different types of columns: Condition Columns and Action Columns. If in a certain row a condition is met, then the according action row is executed. The execution column can again contain the execution of other expressions.



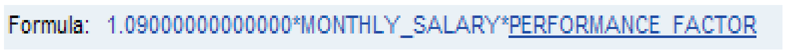
The Decision Table is processed sequentially, therefore the table entries should be arranged starting with the most specific ones, followed by more general ones.

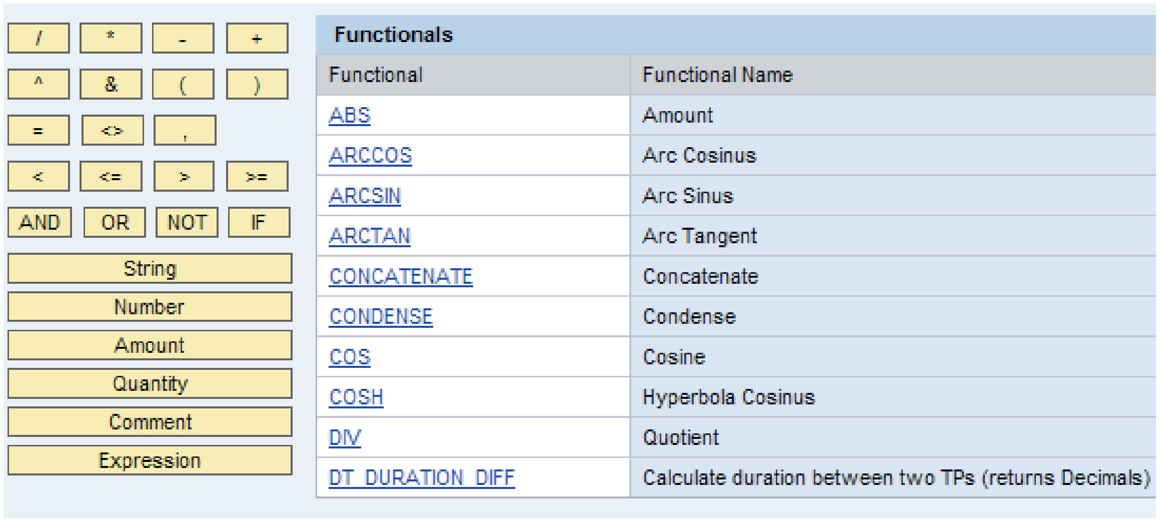
Level of Cover	Age	Set Pricing Factor	Set Base Premium
Normal	[0..29]	1	1000.00000000000000 EUR
Normal	[30..49]	1.2	1000.00000000000000 EUR
Normal	[50..69]	1.5	1000.00000000000000 EUR
Normal	[70..79]	1.9	1000.00000000000000 EUR
Normal	>79	2.4	1000.00000000000000 EUR
Astute	[0..29]	1.5	1200.00000000000000 EUR
Astute	[30..49]	1.8	1300.00000000000000 EUR
Astute	[50..69]	2.25	1400.00000000000000 EUR
Astute	[70..79]	2.85	1600.00000000000000 EUR
Astute	>79	3.6	2000.00000000000000 EUR

Fig. 6.18: A decision table in BRFplus

Formula Expression:

A formula expression makes a calculation based on mathematical formulas. They are a mixture of the values stored in data objects and mathematical expression which can be chosen like displayed in the picture below.


Formula: 1.09000000000000*MONTHLY_SALARY*PERFORMANCE_FACTOR



Functionals

Functional	Functional Name
ABS	Amount
ARCCOS	Arc Cosinus
ARCSIN	Arc Sinus
ARCTAN	Arc Tangent
CONCATENATE	Concatenate
CONDENSE	Condense
COS	Cosine
COSH	Hyperbola Cosinus
DIV	Quotient
DT_DURATION_DIFF	Calculate duration between two TPs (returns Decimals)

Fig. 6.19: A Formula Expression in BRFplus

Because the environment of BRFplus is implemented in SAP's WebDynPro-technology and is not for example an Eclipse-based environment a formula can't be typed using the keyboard. Instead the provided board has to be used (see picture).

BRMS Connector Expression:

The BRMS Connector Expression provides a way to connect the BRMS of another vendor to BRFplus. A detailed description is given in the respective integration chapters.

6.4.3 Testing

Before a function can be used in a production system it should be tested first using the simulation function of BRFplus. Here the values for the input data are defined and the result the function delivers can be discovered.

There are two modes for executing a function in the simulation mode:

- Show Only Result Mode
- Show also Results of Intermediate Steps Mode

The first mode only displays the final result of the execution while the other mode also displays intermediate steps which can be very helpful for debugging.

6.4.4 Rule Engine

There is no official documentation available for the Rule Engine. But according to a statement of the founder of BRFplus, Carsten Ziegler, the Rule Engine uses the Sequential algorithm for rule execution.[26]

6.5 Integration of SAP NetWeaver BRM with SAP BPM

The first scenario which is evaluated is the integration of SAP NetWeaver BRM into a Business Process which has been modeled in SAP NW CE using the Process Composer. Because the Business Process is based on a SOA, the functionality of a ruleset modeled in the BRM is exposed as a Web Service.

The following is assumed:

- The process which will consume the service has already been modeled
- The ruleset which is already modeled and exposed as a web service, i.e. we already have a WSDL file

The process composer of SAP distinguishes between two tasks in a business process, namely "human tasks" and "automated activities". Because Web Services are always bound to an automated activity, the WSDL of the ruleset has to be imported to the process and bound to such an automated activity.

There are three choices of how to fetch the WSDL file:

From

- Enterprise Service Repository
- Remote Location / File System
- Services Registry

PortType:

The interface of the automated activity has to be set to the so called **PortType** of the Service. The PortType of a web service is being generated automatically with the import and represents the interface of the service.

Service Reference:

Also a Service Reference has to be created so that the application server can assign the correct service to the activity. The Service Reference represents the used service and contains the information of how to consume that particular service. It points to the created PortType.

Service Group:

Service References for a certain application are normally grouped to a "Service Group" which makes it easier for administrators to manage the Services for an application when it is necessary, for example when porting from a test system to a production system.

Data Mapping:

Finally a data mapping has to be done between the objects used in the process and the input and output data of the recently web service through the recently assigned interface.

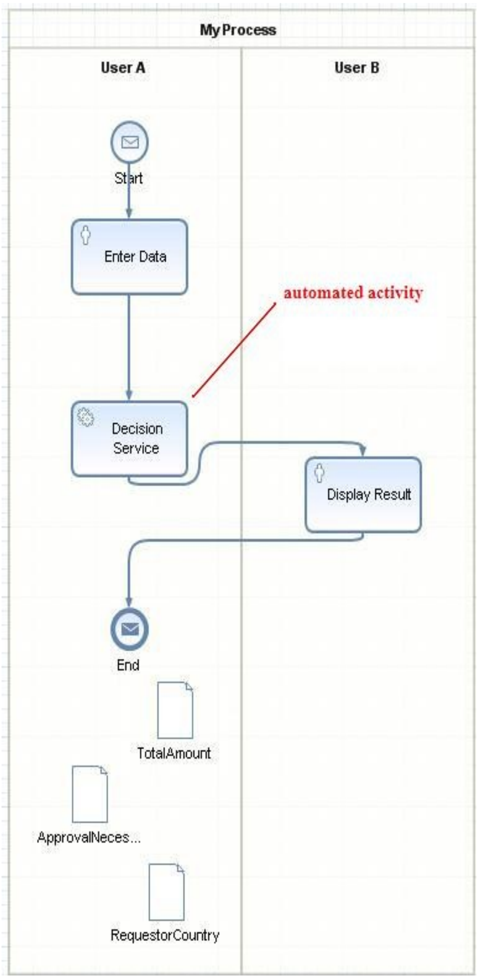


Fig. 6.20: Example of a Business Process modeled using the Process Composer

Configuration steps:

Before being able to the web service in the business process a number of configuration steps have to be done in the administration console of the application server, otherwise the automated activity of the process will not execute.

The following artifacts have to be configured in the Administration Console of the Application Server :

- System Landscape Directory (SLD)
- Service registry
- Provider System
- Communication Profile

– Consumer Proxy (optional)

For the configuration of the SLD and the Service Registry SAP offers a "Wizard-Based Configuration", which sets them up automatically. If the wizard fails, then the set up has to be done manually.

System Landscape Directory:

The System Landscape Directory (SLD) is a repository where all components of an SAP System Landscape are registered, be it hardware components or software components. The description of the components is done using the "Common Information Model" (CIM) and includes for example information about the current version and dependencies between the components. SLD also enables a holistic software-upgrade by SAP for the customer who has defined an own SLD.

Service Registry:

The Service Registry is the UDDI compliant registry of SAP where the web services in the landscape are registered. It contains the WSDL Meta-data and Service-Endpoints of the Service-Providers. With it's help, users, like developers or administrators, can see which services are implemented and ready to use or which have not yet been implemented, amongst other things. A query interface is provided to search for services.

Provider System:

Web Services are always offered by a Provider System for consumption. A Provider System can either be an SAP-System (Java or ABAP) or a Non-SAP System. So whenever a web service is created, it has to be bound to a Provider System which hosts it, so that the calling application or process knows where to connect to.

Communication Profile:

A communication profile handles the configurations concerning the connection to another system, like a provider system. The configurations are for example in the field of security and transaction support. To be able to connect to a provider system it is mandatory to create a communication profile and associate it with the connection. This is done with the creation of the provider system.

Consumer Proxy (optional):

Using a consumer proxy a web service which resides on a provider system can be consumed by the client. The proxy acts like a mediator between the service client and the service provider. Consumer proxies are only used for point-to-point connections between a client and provider.

6.6 Integration of SAP NW BRM with a Java Application

Here it is evaluated how SAP NW BRM is integrated within a Java or Java EE application, where the decision code is externalized to SAP NW BRM.

Because the NW BRM is deployed to the SAP NW Application Server as a stateless session bean the way to integrate NW BRM with the Java or Java EE application is to look-up the services which are offered by this bean through the naming service JNDI.

The integration comprises the following steps:

– Creation of the Rule Engine Object

– JNDI look-up of the stateless Session Bean

– Invoking the ruleset by calling the corresponding method "`invokeRuleset`" and giving the necessary parameters (name of Development Component, name of the ruleset, input objects)

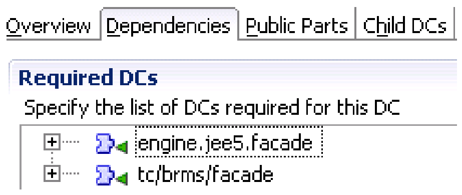
The JNDI name of the Session Bean is called: "`com.sap.brms.RuleEngine`".

The method then returns an object of type `List` with the resulting objects or with exception information in case something went wrong.

Making the Rule Engine available to the Java Development Component:

To be able to create the Rule Engine in the application it has to be introduced to the Java Development Component. This is done by adding a dependency from the Java DC to a DC called "BRMS-Facade".

Generally a Facade provides an API for certain functionality, which in this case is the Rule Engine of SAP NW BRM.



Required DCs

Specify the list of DCs required for this DC

engine.jee5.facade

tc/brms/facade

Fig. 6.21: A dependency to “tc/brms/facade” enables to use the Rule Engine

6.7 Integration of SAP NW BRM with an ABAP application

As mentioned in the BRFplus chapter, an ABAP application calls BRFplus through six lines of Code which can be checked in the Appendix.

The integration of an ABAP application with SAP NW BRM is accomplished by using BRFplus as a mediator. For this purpose the expression type called "BRMS Connector" has been implemented in BRFplus. The BRMS Connector not only enables the mediation to SAP NW BRM, but to any external BRMS.

To connect BRFplus with NW BRM the expression type "BRMS Connector" has to be defined as the top expression of a function, which runs in Functional Mode. Now when an ABAP application calls this function the functionality of the BRMS Connector is executed and the connection will be established.

The communication between BRFplus and NW BRM is based on the following SAP constructs:

Remote Function Call (RFC):

The Remote Function Call is very similar to the known "Remote Procedure Call", because it provides the capability of calling a remote function. It provides an interface to applications written in ABAP and enables the communication to these applications from other applications outside of the ABAP application server.

Function Module:

Function Modules are pre-defined procedures written in ABAP that can be called by any ABAP application. They can be accessed remotely by defining a so called "RFC-Destination", which contains connection parameters for establishing an RFC.

Java Connector (JCO):

The Java Connector is a middleware component of SAP which enables the communication between components written in ABAP and Java. The communication is possible in both directions:

– An ABAP component be called by a Java Application (Inbound Call)

– A Java Application can be called by an ABAP application (Outbound Call)

To use the Java connector as an Outbound call a JCO Server has to be implemented on the Java System, which handles incoming request from ABAP applications. In case a

Java Application, which is deployed on the SAP NetWeaver Application Server, there exists an already integrated JCO Server in the NetWeaver AS. Otherwise a standalone JCO Server has to be implemented.

The Java Connector has a repository with a set of function modules to fulfill it's tasks. These function modules are then implemented in the JCO Server program. JCO also provides some useful functionality like Connection Pooling.

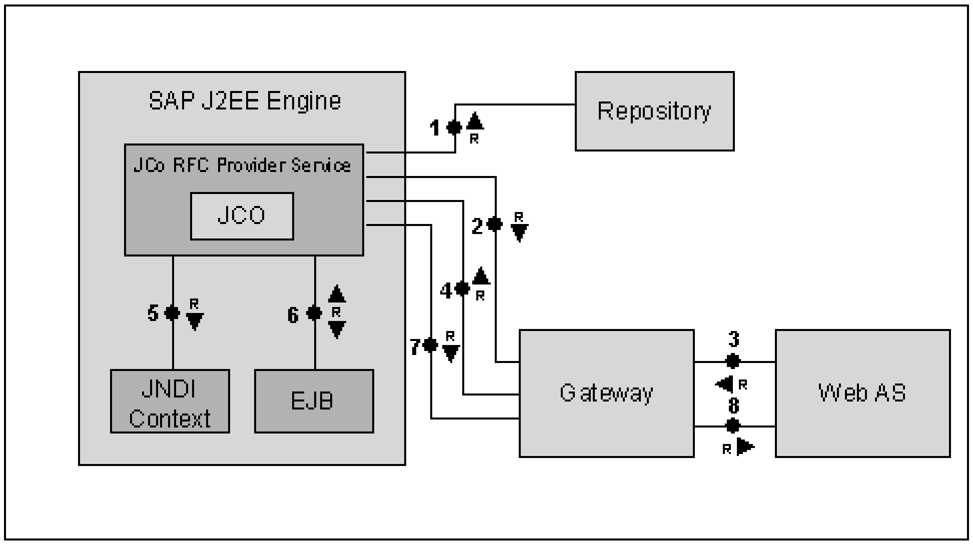


Fig.6.22: Communication of SAP's ABAS AS with JAVA AS through the Java Connector [27]

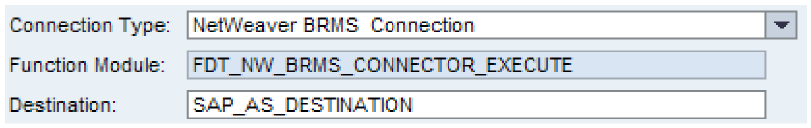
This picture illustrates the communication steps between an ABAP application deployed on the ABAP AS (called Web AS in the picture) and a Java application deployed on SAP Java AS through a JCO Server. The built-in JCO Server in the SAP Java AS is called "JCO RFC Provider Service".

The following steps are necessary for an Outbound Call:

1. The JCO Server looks up the desired function module in the repository and provides and implementation (by the developer).
2. The JCO Server binds itself to an SAP "Gateway" as a so called "JCO Program".
3. An RFC Destination is defined in the ABAP AS, which points to the JCO Program at the Gateway.

4. The JCO Server receives an incoming request from an ABAP application through the Gateway.
5. The implementation of the function module is discovered through the JNDI Naming Service of the Application Server.
6. The EJB which contains the implementation of the function module generates a result and sends it back to the application server.
7. The JCO Server sends the result to the Gateway
8. The Gateway sends the result to the ABAP AS, where the ABAP application in deployed

For the integration of BRFplus with the SAP Java AS, the JCO function module which needs to be implemented in the EJB is called `FDT_NW_BRMS_CONNECTOR_EXECUTE`, which is automatically selected in the BRMS Connector.



Connection Type:	NetWeaver BRMS Connection
Function Module:	FDT_NW_BRMS_CONNECTOR_EXECUTE
Destination:	SAP_AS_DESTINATION

Fig. 6.23: The BRMS Connector uses a special Function Module

In this case the RFC Destination is called "`SAP_AS_DESTINATION`" and has to be defined using the Administration Console of the ABAP AS.

Signature of the BRMS Connector:

The parameters of the signature of the BRMS Connector have to be defined. They can be one of the two types:

– **Exporting:** Parameters of type Exporting are read-only and cannot be changed by the external BRMS. Even if the external BRMS changes them these changes are not reflected in BRFplus.

– **Changing:** Parameters of type Changing on the other hand are allowed to be changed by the external BRMS.

Passing the data objects:

The data objects are passed in an XML file to the external BRMS and also the response which BRFplus receives comes in form of an XML file. The XML file is not sent as plain text, instead it is coded in the Binary64-Format, also known as a Byte-Array in Java. A Byte-Array in SAP is called "`XSTRING`", which has nothing to do with the ordinary String type in Java.

Another information which can be sent to SAP NW BRM is the specific rule DC and ruleset is going to be called. This information is of course necessary to be able to know which ruleset to call, but it is not mandatory to send this information together with the data objects. This is due to the fact, that the ruleset will ultimately be called in the JCO Server and the programmer can also specify the ruleset there, without the information of BRFplus. However, if you some automation is intended then the information of the rule project and the ruleset should be sent and extracted in the JCO Server and then automatically be executed.

The function module has the following structure:

Type	Name	SAP Data Type	Remarks
`IMPORTING`	`iv_input_xml`	`XSTRING` (= byte array)	Context XML passed to BRMS
`IMPORTING`	`its_call_parameter`	`fdtt_brms_call_param`	BRMS specific parameters needed for execution
`IMPORTING`	`iv_date_time_iso`	`/ISD/DATE_ISO`	Date time for executing rules that were active at some point in time
`EXPORTING`	`ev_output_xml`	`XSTRING`	Modified XML returned by BRMS

Table 6.1: Structure of Function Module `FDT_NW_BRMS_CONNECTOR_EXECUTE`

The Objects of type "`IMPORTING`" are the ones which are sent by BRFplus and received by the function module.

The Objects of type "`EXPORTING`" are sent back by the module to BRFplus.

The mandatory objects to make a call are "`iv_input_xml`" and "`ev_output_xml`".

– `iv_input_xml` is the XML which contains the input data for SAP NW BRM.

- `ev_output_xml` is the XML which contains the result and will be sent back to BRFplus.
- `its_call_parameter` contains the name of the rule project and the ruleset which will be called. Since this information can also be hard-coded in the JCO Server this is optional, nevertheless it's not a good practice.
- `Iv_date_time_iso` contains information about the time for executing the ruleset, which is optional and rarely used.

To be able to model the rules in NetWeaver BRM using the data objects defined in BRFplus the functionality is provided by BRFplus to generate an XML Schema containing the data objects. This XML Schema has to be imported in NetWeaver BRM, so that it can be verbalized and used for the creation of the ruleset.

7 Business Rules Interoperability Scenarios

In this chapter it is evaluated, how the BRMS of IBM and SAP which have been described in the preceding chapters can inter-operate in a cross-vendor scenario. This includes the integration between the BRMS themselves and the integration of IBM's BRMS with the ecosystem of SAP. The other way, i.e. the interoperability of SAP's BRMS with IBM's ecosystem is not within the scope of this thesis.

7.1 Integration techniques

The interoperability scenarios which are discussed in this chapter focus on two things:

– How to leverage rules modeled in JRules in SAP's BRMS

– How to leverage rules modeled in JRules in SAP's ecosystem, i.e. process or application in SAP

The products of IBM and SAP, which are involved in the interoperability scenarios are:

IBM:

– IBM WebSphere ILOG JRules

SAP:

– SAP NetWeaver BRM

– SAP BRFplus

– SAP NetWeaver Composition Environment

– SAP NetWeaver Application Server

The following approaches for integration are considered:

– Web Service

– Java EE integration

– Pass-Through Rule

– Export / Import

Web Service:

The BRMS of IBM and SAP offer the functionality of exposing their rulesets as a web service, therefore this approach aims to consume these web services in an application or a business process.

Java EE integration:

Since the BRMS of IBM and SAP are enterprise applications, this method targets the integration of the rule engine in an Application Server so that it can be called by any Java EE application using an Enterprise Java Bean. This Java Bean is either a stateless or a stateful Session Bean.

Pass-Through Rule:

As we have seen BRFplus offers an expression type called "BRMS Connector". With it's help rules can be defined in BRFplus but executed in another BRMS, while this is completely transparent to the calling ABAP application who just calls the function defined in BRFplus. We hereafter call this kind of rule, which passes incoming calls to another BRMS as a "Pass-Through" rule.

Export/Import:

With Export/Import we consider an interoperability Scenario between different BRMS, where the Rules in one BRMS are exported in one BRMS and imported in the other one.

This method has two basic requirements:

– The functionality to export or import rules must be provided by both BRMS.
– There must be a translation between the two BRMS, where the semantics of the rule languages must be preserved.

A2R and R2R scenarios:

We hereafter call the integration of a BRMS with another BRMS "Rule to Rule"-Interoperability (R2R) and the integration of a BRMS with an application or business process "Application to Rule"-Interoperability (A2R).

The following two tables give an overview about the interoperability scenarios discussed in this thesis, separated in A2R and R2R. It contains interoperability scenarios within a specific vendor, SAP or IBM, as well as the cross-vendor interoperability scenarios.

While the interoperability scenarios within a specific vendor have been discussed in the preceding chapters, the cross-vendor scenarios, which are marked in green, will be discussed in detail.

A2R scenarios	IBM WebSphere ILOG JRules	SAP NetWeaver BRM	SAP BRFplus
IBM WebSphere Application Server	Stateless/Stateful Session Bean	(NIS)	(NIS)
IBM WebSphere Integration Developer	Web Service (HTDS), SCA Component	(NIS)	(NIS)
SAP NetWeaver Composition Environment	**Web Service**	Web Service	Web Service
SAP NetWeaver Application Server	**Java EE Integration**	Stateless Session Bean	Web Service
SAP ABAP Application	**(Web Service)***	(Web Service)	6 Lines of ABAP Code

Table 7.1.: Scenarios for "Application to Rule Interoperability" (A2R)

* It is possible make a Web Service call directly from an ABAP application, therefore it is also possible to call JRules from an ABAP application, since a ruleset in JRules can be exposed as a Web Service. However, applications written in the ABAP-language are not evaluated in this thesis.

R2R scenarios	IBM WebSphere ILOG JRules	SAP NetWeaver BRM	SAP BRFplus
IBM WebSphere ILOG JRules	X	(NIS)	(NIS)
SAP NetWeaver BRM	**Export/Import, Pass-Through Rule**	X	(NIS)
SAP BRFplus	**Pass-Through Rule**	Pass-Through Rule	X

Table 7.2.: Scenarios for "Rule to Rule Interoperability" (R2R)

The fields with the entry "NIS" (not in scope) indicate that there might be some interoperability potential, but are not in the scope of this thesis, since they are about the integration of SAP products in the IBM environment.

7.2 Integration of JRules with SAP NetWeaver CE (A2R)

The integration of JRules with NetWeaver CE can be generally achieved in two possible ways:

- Using the Web Service approach
- Using a Java EE integration

7.2.1 Web Service Approach:

Since SAP NetWeaver is a platform which supports the building of an IT Landscape using the SOA approach, it should be possible to integrate JRules in a business process modeled in NetWeaver CE. One of the main advantages of SOA is that the integration of a third party system into an IT landscape is enabled through the web service standards no matter what platform the third party system has been built on.

So since the rulesets in JRules can be exposed as web services using the Hosted Transparent Decision Service (See JRules Chapter for details), JRules will be integrated in a business process modeled with the "Process Composer" of NetWeaver CE.

The integration of SAP NW BRM in a business process modeled in SAP NetWeaver CE has been accomplished in the previous chapter. The integration of ruleset modeled in JRules in the business process is a similar task with some differences in the configuration steps.

The following situation is assumed:

- A business process has already been modeled using the Process Composer
- A RuleApp which contains at least one ruleset has been modeled in JRules and has been deployed to the Rule Execution Server
- The service landscape directory (SLD) has already been configured in the SAP AS Administration Console

To expose the functionality of the desired ruleset as a web service the corresponding WSDL-file has to be fetched, which is offered through the Hosted Transparent Decision Service (HTDS).

To be able to integrate this web service in the business process an "automated activity" has to be created at the right place in the workflow.

In the project containing the business process the WSDL file of the web service then has to be imported which automatically generates the interface to the service and the port type which points to the service. A service reference then has to be created which resides in an automatically generated service group – or put in an already existing service group.

The intention is now to create a point to point connection between the business process modeled in SAP and the Web Service of the ruleset modeled in JRules, which requires the following steps:

Creation of a Connection in the SAP Admin Console:

Each Web Service which is going to be used in the business process has to be bound to a Provider System, which hosts the Web Service. In this case the Provider System is an application server which the rule execution server has been installed on, e.g. WebSphere Application Server. Also a Communication Profile has to be associated to this provider system, which handles the security and transaction policies of the connection (See the previous chapter for the definition of a Communication Provider and Provider System). The default Communication Profiles can be used for testing purposes. The communication with the Provider System is done through a Communication Proxy, which has to be created. After the deployment of the business process on the application server, the JRules Web Service has to be explicitly assigned to the business process as a “Consumed Service”.

Now the business process modeled in SAP consumes the functionality of a ruleset modeled in JRules exposed as a web service.

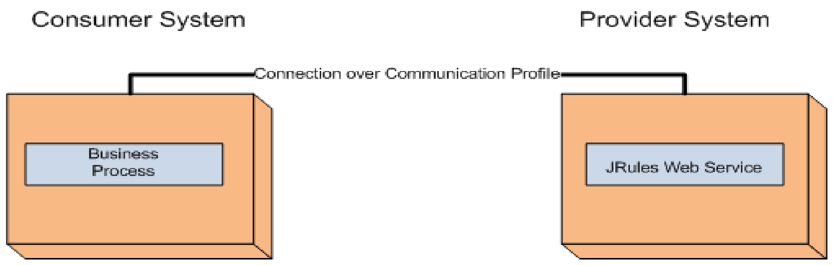


Fig.7.1.: JRules Web Service is hosted on a Provider System and is accessed through a Communication Profile

7.2.2 Java EE Integration

This approach evaluates the possibility to leverage JRules as Java EE add-on for the SAP NetWeaver Application Server.

The Java EE specification defines a standard way to deploy an enterprise application to an application server. Both Application Servers of SAP and IBM are Java EE certified and the BRMS of both vendors are Enterprise Applications which are deployed on these application servers. So the BRMS of one vendor should be deployable on the Application Server of the other vendor, or, regarding our interoperability scenarios, JRules should be deployable on SAP NW AS. This would enable any Enterprise Application deployed on SAP NW AS to use the rule engine of JRules through a Session Bean.

But companies like IBM and SAP always enhance their Java EE certified application servers which additional functionality to keep an advance towards other vendors of application servers. If they didn't do that and would just stick to the Java EE specifications without any enhancements, customers wouldn't pay any money for them, because there are already Java EE certified open source application servers, which are free to use.

So, after the individual modifications of the application servers they are still Java EE certified, but the full interoperability of applications with regards to their deployment on Java EE certified application servers is not given anymore. This is why vendors of Java

EE applications always have a list of application servers which are supported by the vendor, and the major application servers in the market are normally supported.

However, SAP's application server is rarely used as a standalone application server outside of the NetWeaver context and has a very low market share. This is a reason, why IBM does not support the Java EE integration of JRules in the SAP NetWeaver application server, therefore it can't be found on the list of supported application servers.

In Version JRules V. 7.1 IBM supports and provides installation instructions for the following Application Servers:

- WebSphere Application Server / WebSphere Community Edition
- Tomcat
- JBoss
- WebLogic
- Sun Application Server

7.3 Integration of JRules with SAP BRFplus (R2R)

Using the functionality of a BRMS in conjunction with an ABAP application is generally done with BRFplus. The construction of rules and it's management in BRFplus is not that powerful and intuitive as with an Eclipse-based BRMS with an extra environment for Business Users like SAP NW BRM or JRules.

But BRFplus offers an expression type called "BRMS Connector" with which an external BRMS can be connected to BRFplus so that the rules which have been modeled and executed in the external BRMS are used by BRFplus which again sends the results to the ABAP application. So BRFplus offers the functionality to act as a mediator between the ABAP application and the external BRMS.

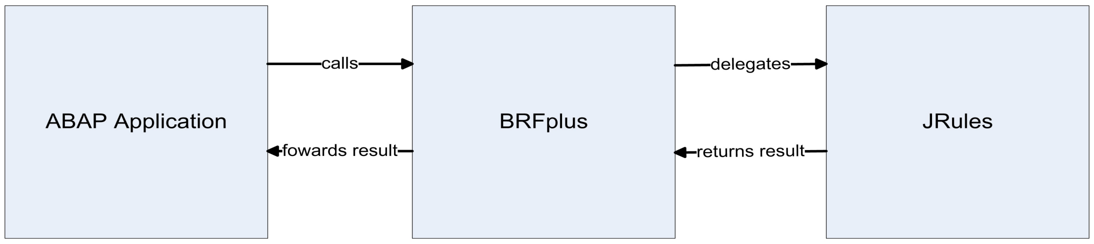


Fig. 7.2: BRFplus as mediator between an ABAP application and JRules

The BRMS Connector expression type has already been used in the previous chapter to connect BRFplus to SAP NetWeaver BRM. Here it will be used to connect to IBM's BRMS ILOG JRules which makes it possible to use the rules modeled in JRules in an SAP ABAP application.

Here are the steps, which have to be done in BRFplus:

- In BRFplus a function is created which uses the expression type "BRMS Connector" as the top expression
- The signature of the BRMS Connector is defined so that it knows which objects to hand over to JRules
- An RFC Destination is created using the Administration Console of the ABAP based Application Server of SAP Business Suite
- The name of the RFC Destination and the Function Module is specified in BRFplus. The name of the Function Module is `"FDT_EXT_CONNECTOR_EXECUTE".`
- After the signature of the BRMS Connector has been defined, the XML Schema must be generated, which contains the definition of the data types used in the signature so that it can be imported in the external BRMS and worked with. The Schema will be the Execution Object Model in JRules (XOM).

Here are the steps, which have to be done in JRules:

- The rules project is created in JRules and the Schema is imported as the XOM.
- The Rules are modeled.
- The Ruleset are deployed to the Rule Execution Server and exposed as a web service through HTDS.

After modeling the necessary rules in BRFplus and JRules a JCO Server is needed on the Java system to handle the incoming ABAP request and call JRules.

There are two ways to build a JCO Server for this purpose. The first option is to download the standalone JCO-Server offered by SAP and to develop the code to handle the incoming request. The second option is to use IBM's "SAP Adapter" in the WebSphere Integration Developer, which is deployed as a Java EE Resource Adapter and has an integrated JCO Server to handle connections to ABAP applications. Both options are described in the following.

7.3.1 Using a Standalone JCO Server

After installing the JCO Server and making the necessary configurations the programming of the JCO Server can be started, which consists of registering the Server at the SAP Gateway and afterwards implementing the business logic for calling JRules.

All classes provided by the JCO-library are inner classes of the main JCO class. The JCO class provides static methods to instantiate these classes. The basic methods needed for the initial configuration of the JCO Server are the following:

- `JCO.addClientPool:` Creates a connection pool for handling SAP-Requests. It is also possible to create direct connections using `JCO.createClient()`, but to be able to handle lots of requests, the creation of a pool is recommended due to performance issues with lots of direct connections. The information about the connection to the SAP-Gateway have to be provided to this method.
- `JCO.createRepository:` Using the Client-Pool this method creates a repository which contains the templates for the SAP function modules. Since the access to incoming data is always done through a function module, it is crucial for the JCO Server to own the templates.
- `JCO.Server:` This class is the most important class, because it represents the listener which registers itself at the gateway as a JCO-Program. An RFC Call from SAP always targets this program.

The business logic is implemented by implementing the `JCO.Server` class – more precisely the method `handlerequest(JCO.Function function)` inside the class, where `JCO.Function` represents the interface to a function module taken from the repository.

To be able to retrieve the data which is sent using a function module the name of the function module has to be known and it's structure.

The function module used by BRFplus to send data to JRules is called:

`FDT_EXT_BRMS_CONNECTOR_EXECUTE`, which has the same structure as `FDT_NW_BRMS_CONNECTOR_EXECUTE`, which is used for SAP NW BRM.

Type	**Name**	**SAP Data Type**	**Remarks**
`IMPORTING`	`iv_input_xml`	`XSTRING` (= byte array)	Context XML passed to BRMS
`IMPORTING`	`its_call_parameter`	`fdtt_brms_call_param`	BRMS specific parameters needed for execution
`IMPORTING`	`iv_date_time_iso`	`/ISD/DATE_ISO`	Date time for executing rules that were active at some point in time
`EXPORTING`	`ev_output_xml`	`XSTRING`	Modified XML returned by BRMS

Table 7.3: Structure of the Function Module "`FDT_EXT_BRMS_CONNECTOR_EXECUTE`"

The mandatory objects to make a JRules call are "`iv_input_xml`" and "`ev_output_xml`".

– `iv_input_xml` contains the string where input data for the JRules ruleset resides.

– `ev_output_xml` is the XML which contains the result and will be sent back to BRFplus.

– `its_call_parameter` contains the name of the rule project and the ruleset which will be called. Since this information can also be hard-coded in the JCO Server this is optional, nevertheless it's not a good practice.

– `iv_date_time_iso` contains information about the time for executing the ruleset, which is optional and rarely used.

After fetching the input data from the XML-File using an XML-Parser (preferably a DOM-Parser since `ev_output_xml` needs to be created with it) the ruleset modeled in JRules can be called. This is done through a Web Service call using for example "Axis" from Apache.

After receiving the result it has to be integrated in the XML-File `ev_output_xml` which will be then sent back to BRFplus as a result of the RFC call.

7.3.2 Using the SAP Adapter in WID

When installing the WID there is the option to install a so called "SAP Adapter", which installs in the Process Server as a Java EE Resource Adapter. An Adapter in WID is represented as an SCA component and offers the functionality to connect to an external System. This SCA component can then be connected to other components like any other SCA component.

The SAP Adapter in WID is similar to the JCO RFC Provider in the Administration Console of the SAP Application Server. It offers an integrated JCO Server which handles the connection to the ABAP Application Server and registers the JCO Program at the SAP gateway.

Here is the Assembly Diagram modeled in WID:

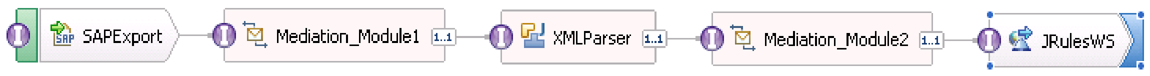


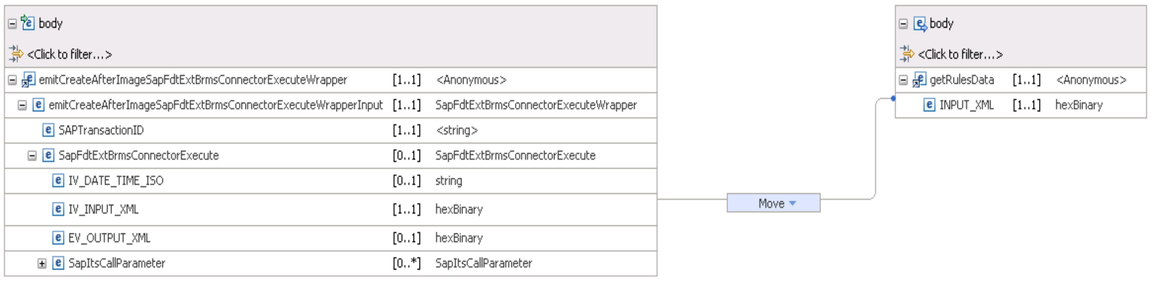
Fig. 7.3: Assembly Diagram for the connection of BRFplus with JRules

<u>SAPExport:</u>

"SAPExport" is the mentioned SAP Adapter, which contains an integrated JCO Server. The configurations, which are necessary for a successful connection have to be made and desired Security Settings have to be set here.

<u>Mediation_Module1:</u>

This is the first mediation module which is used in this scenario. It has the task of getting the Base64-encrypted XML-File from BRFplus, which contains the Input data for JRules and passing it to the Java Application, which is connected to the other end of the mediation module.



body
<Click to filter…>

emitCreateAfterImageSapFdtExtBrmsConnectorExecuteWrapper	[1..1]	<Anonymous>
emitCreateAfterImageSapFdtExtBrmsConnectorExecuteWrapperInput	[1..1]	SapFdtExtBrmsConnectorExecuteWrapper
SAPTransactionID	[1..1]	<string>
SapFdtExtBrmsConnectorExecute	[0..1]	SapFdtExtBrmsConnectorExecute
IV_DATE_TIME_ISO	[0..1]	string
IV_INPUT_XML	[1..1]	hexBinary
EV_OUTPUT_XML	[0..1]	hexBinary
SapItsCallParameter	[0..*]	SapItsCallParameter

Move

body
<Click to filter…>

getRulesData	[1..1]	<Anonymous>
INPUT_XML	[1..1]	hexBinary

Fig. 7.4: Mediation Module passing the XML file with the input data

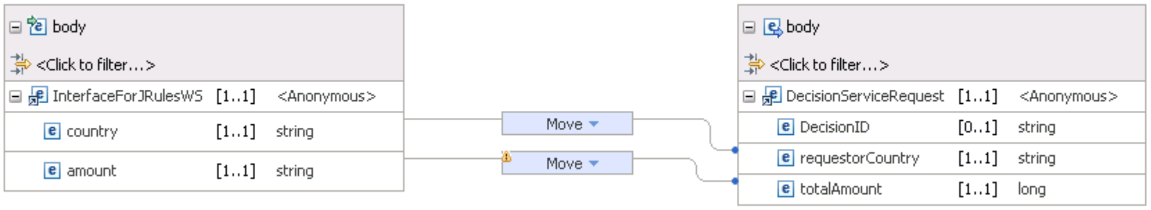
While in this example only the mandatory XML-File stored in the variable `"IV_INPUT_XML"` has been passed, it is also possible to pass the optional variables, like `"SapItsCallParameter"`, which contains the desired Rule project and ruleset.

XMLParser:

This is a Java Component, which mainly consists of the DOM-Parser, which extracts the input values contained in the XML file, which has been passed through the mediation module. It also contains the logic to call the JRules Web Service through the next mediation module using the extracted data.

MediationModule2:

The second mediation module passes the Web Services call from the preceding Java Application to JRules. For that, it has to make a mapping between the values, which have been extracted from the XML to the parameters, which are expected by the Web Service and is specified in the WSDL file.



body
<Click to filter…>

InterfaceForJRulesWS	[1..1]	<Anonymous>
country	[1..1]	string
amount	[1..1]	string

Move
Move

body
<Click to filter…>

DecisionServiceRequest	[1..1]	<Anonymous>
DecisionID	[0..1]	string
requestorCountry	[1..1]	string
totalAmount	[1..1]	long

Fig. 7.5: Mediation Module mapping the data from Java to the Web Service Interface

JRulesWS:

This module represents the Web Service Port, through which the Web Service provided by JRules can be called. Prior to that, the WSDL file, which has been generated in JRules using HTDS has to be imported. The interface representing the required input parameters and operation is generated automatically .

7.4 Integration of JRules with SAP NetWeaver BRM (R2R)

7.4.1 SAP NW BRM Pass-Through Rule:

The idea behind a Pass-Through Rule in SAP NetWeaver BRM is to model a rules, which can be called by an application or process in SAP like any other rule. The rule get's input data and delivers output data to the caller. But instead of calculating the result itself, SAP NW BRM acts as a broker and sends the input data to JRules where the rules which calculate the result are modeled. JRules then sends a result to SAP NW BRM and this is again passed to the calling application or process, for whom this process is completely transparent, i.e. it does not know that the result was calculated by JRules.

SAP NW BRM would have to use a Web Service call to pass the input data to JRules or make a remote invocation of the Session Bean which calls the JRules Rule Engine. But there is no way to implement that logic in a rule, for example using Java Code.

Therefore the modeling of a Pass-Through Rule in SAP NW BRM is not a feasible option for an interoperability scenario.

The functionality in BRFplus has been implemented with the BRMS itself and is not a plug-in. So the intention to pass rules to another BRMS has been defined during the development of BRFplus. This has not been the case with SAP NW BRM.

7.4.2 JRules Export, SAP Import:

In this scenario it is evaluated, if it is possible to export Rules modeled in JRules and import them in SAP NetWeaver BRM.

JRules indeed offers the functionality to export Rulesets into a so called "Ruleset Archive" to be able to import them for a later use or for exchange with another JRules instance. So the requirement of exporting rulesets is fulfilled.

The problem in this scenario is that in SAP NW BRM the option to import rulesets does not exist, because it has not been implemented yet by the developers. Without being able to import rulesets the interoperability between SAP NW BRM and JRules using the import/export method fails.

Also it is not known what the technical rule language the rule engine works on. The developer or business user models the rules against the BOM using a business rule language. The BRL is then internally translated into a technical language which the Rule Engine Executes the rules on, but there is no information provided by SAP how the syntax of this language looks like.

If there was an import functionality in SAP NW BRM and the technical rule language would be known, then the exported ruleset from JRules could be translated from the ILOG Rule Language (IRL), which is the language with which the JRules Rule Engine works, to the language, with which SAP's Rule Engine works.

Nevertheless the interoperability between BRMS of different vendors is of the industry's interest. It is also a desire to create a standard rule language which can be executed on any BRMS which is compliant to that standard. Because there is a lot of progress in this field which might lead SAP and IBM to adhere to that standard in the near future, the next chapter discusses this topic.

7.5 Approaches to exchange Rules between different BRMS

As we have seen, the export/import option is not a feasible interoperability option.

But the industry is interested in the creation of a standard rule language where rules can be modeled in and consequently be executed in every BRMS which conforms to a certain standard which is defined along with defining the standard rule language.

Especially the organization W3C has been working on ways to enable rule interoperability between different BRMS. The approaches and the results achieved so far are the topic of this chapter.

One of the motivations to create a standard lies in the creation of the so called "Semantic Web" which is a goal of the W3C to achieve. A standard would also enable to introduce rules already in the design phase of the development of an application, like in the definition of UML diagrams.

7.5.1 Semantic Web

The Semantic Web which is also called "Web 3.0" is going to be next evolutional form of the World Wide Web. Right now it is possible to interchange data between computers through the infrastructure of the internet, but the meaning of the data, i.e. the information, can only be understood by a human.

But the huge and ever increasing amount of data flowing through the internet raises the desire to make this data understandable for computers, because then the information can be used more precisely and effectively due to the fast calculation capability of computers. For example if a person says that it wants to go to holidays at a certain place the computer can then offer additional information to this person tailored to his desires and habits.

A way to make web pages understandable for computers is to enhance the them with annotations from a language called "Resource Description Framework" (RDF) [28].

The two noticeable approaches for Rules standardization and Rules Exchange are the "Rule Interchange Framework" (RIF) and the "Rule Markup Language" (RuleML).

7.5.2 Rule Interchange Format (RIF):

The goal of the Rule Interchange Framework is to offer an XML-based syntax with which different BRMS can interchange their rules in a semantics-preserving way. So when a BRMS wants to exchange rules with another BRMS it would first export them into an XML File in RIF and send them over to the other BRMS which in return imports the RIF XML to its own specific language with which it's rule engine can work. Of course it is also theoretically possible for a rule engine of a BRMS to directly work with the Rule Interchange Format, which would then make RIF a standard language for modeling rules. [29]

It is a nearly impossible challenge to create a standard which preserves the semantics of every Rule Language on the Market, because rules can have been created for completely different purposes, and therefore the syntax and semantics of the rule languages differ. Rule languages can be based on first order logic which do not have an action-part. Other rule languages are based on "production rules" which have the ability to negate or retract facts in the working memory and are used in a BRMS – to name two examples. This is why RIF has introduced the concept of special dialects which define common syntax and semantic for each type of rule language. This means that Rule Systems designed for the same purpose use the same RIF-dialect for rule exchange.

As mentioned, the rule language of a BRMS is based on Production Rules. The RIF-dialect for Production Rules is called "RIF-PRD" (RIF Production Rules Dialect).

So, in an Export/Import-Scenario between BRMS the exported rules are translated into RIF-PRD and sent to to the target BRMS. There the rules in RIF-PRD will be translated back to the proprietary Rule Language and executed. However, the dialect does not yet cover all functionality of the proprietary Rule Languages, but there is ongoing work on extending the capabilities and reach of the RIF dialects.

Since June 2010 RIF is a W3C Recommendation and first tests of exchanging simple rules between three different BRMS (ILOG JRules, Oracle BRMS and Prova) using RIF have been successful [30].

7.5.3 Rule Markup Language (RuleML):

Another initiative which shall be mentioned in this thesis for creating a standard Rule Language is the Rule Markup Language. It's purpose is to serve as a standard rule language for the semantic web with which rules can be modeled and executed on every Rule Engine which conforms to the standard. This would also facilitate the approach of Model Driven Engineering (MDE) because after modeling the Rules the code generation in the specific system could be automated and therefore the Software Development Process would be accelerated. An important aspect for achieving this goal is to have a common API to access a rule engine, which is realized with JSR-94.[31]

7.5.4 JSR-94:

JSR-94 is a Java Specification Request for a common API to access a Java Rule Engine. With it's help any Java application, be it a standard Java Application developed in Java SE, or an enterprise Java Application developed in Java EE, can access any rule engine, which implements this specification. As a consequence replacing JSR-94 compliant BRMS in an application server does not involve any changes in the applications which make use of the rule engine.

While IBM JRules is compliant to JSR-94, there is no information about SAP NW BRM. [32]

7.6 Opinion on R2R results

In this chapter it has been evaluated in which ways an interoperability between IBM's BRMS JRules and SAP's ecosystem, including their BRMS, can be established. The integration of JRules with a Business Process or application written in SAP was part of the "Application to Rule"-Interoperability and the integration of JRules with SAP NW BRM or BRFplus was part of the "Rule to Rule"-Interoperability.

The existing interoperability between the Enterprise Portals of IBM and SAP, which was briefly described in chapter 2, served as a pattern, therefore the Import/Export scenario was equal to the "Portal-Hub" approach and the Pass-Through-Rule scenario was equal to the "Portal-in-Portal" approach.

The interoperability with SAP NW BRM was of high interest, because SAP is moving from the proprietary ABAP-world to the Java-world, which generally enhances the interoperability with other applications – therefore SAP NW BRM remains the main BRMS of SAP. However, to enable a direct interoperability with another application, the software has to reach a certain degree of maturity and SAP NW BRM is a rather new appearance on the BRMS market, which relates to SAP's entry in the BPM arena.

While JRules enables the basic requirements for interoperability through providing the possibility to export modeled rules and exposing the information of the technical language, which the rule engine works on, these requirements are not yet provided by SAP. My guess is, that SAP is putting first a higher priority on enhancing the functionality of NW BRM by adding more features related to rules modeling, which will be available with each release of SAP NetWeaver Composition Environment. The necessity of exposing the technical Rule Language or providing an import/export feature is not there yet, due to a low priority. However, it is possible, that in the future SAP will force the compliance of SAP NW BRM to official standards for Rule exchange between different BRMS, like RIF, and to achieve that, it will have to provide the mentioned requirements.

Also, if these requirements are fulfilled, then a possible interoperability through W3C standards like RIF should be considered for evaluation, rather than constructing an own transformation method between the rule languages of JRules and SAP NW BRM.

Nevertheless, SAP's ABAP based BRMS is also evolving and being enhanced with new features, therefore it is pleasant, that the pass-through rule is an already integrated feature in BRFplus. But it is also logical, that this feature has definitely a higher priority for BRFplus than for SAP NW BRM, because of the high difference in the modeling environments, namely WebDynpro for BRFplus and Eclipse for SAP NW BRM or JRules. The modeling of rules in Eclipse is in my opinion much more intuitive and quicker possible than in a WebDynpro-based User Interface.

8 Summary and Outlook

As mentioned in the introduction, this thesis consisted of three tasks, which will be provided a summary of:

1. Exploring the general functionality of the BRMS offered by IBM and SAP
2. Describing how the BRMS of IBM and SAP are embedded in their respective ecosystem
3. Evaluating the interoperability aspects between these systems, with focus on the integration of IBM's BRMS in SAP's system (BRMS and ecosystem). Also elaborating to which extent such an interoperability is feasible.

Exploring the general functionality of the BRMS offered by IBM and SAP:

The BRMS offered by IBM is called JRules and is Java based. SAP offers two BRMS, one is ABAP based and one is Java based. The ABAP based BRMS is called BRFplus and is part the SAP NetWeaver Business Suite. It enables ABAP applications to make use of the advantages offered by a BRMS through sourcing the decision logic of the application out to BRFplus. BRFplus provides an interface based on SAP's WebDynpro technology and enables the modeling of business rules through the basic constructs, which are mainly if-then rules and decision tables. The richness of functionality is not as high as most other BRMS and the rule engine is not configurable and doesn't use the Rete algorithm for pattern matching. Nevertheless in my opinion it provides a well possibility for ABAP programmers to benefit from a BRMS.
SAP NetWeaver BRM is SAP's Java based BRMS and is part of SAP NetWeaver Composition Environment, which is SAP's BPM tool. Because this BRMS is based on Eclipse it enables a far more intuitive rule modeling than BRFplus. While in it's initial release with SAP Composition Environment 7.1 it was missing a lot of desirable features for a BRMS, like the modeling of rules using a rule flow, with version 7.2 a lot of features have been added which makes SAP NW BRM a competitive BRMS in the market. More features can be expected for 7.3.
IBM's BRMS JRules originates in the acquisition of ILOG in 2006, which offered a fully developed market leading BRMS. Therefore it was expected, that JRules would provide the highest amount of features among these 3 BRMS. Indeed this is the case, which is most noticeable when looking at the modeling environment for business users and the configuration options for the rule engine. JRules also offers a by far higher amount of

functionality than it is covered in this thesis, but even the evaluation of the basic features approves, that compared to SAP's BRMS it is the most advanced.

Describing how the BRMS of IBM and SAP are embedded in their respective ecosystem:

In this part the BRMS of IBM and SAP were embedded in the ecosystem of their vendor. Because today companies are often adopting the concepts of Service Oriented Architecture for their IT landscape, BRMS play a valuable role in the business processes of company by taking care of the decision logic, which can be reused across multiple processes.
Therefore the main focus was on exposing a modeled ruleset as a service and then embedding it in a process which has been modeled using the company's BPM tool, while also evaluating, how the BRMS can be called from a Java- or ABAP-application.
Both IBM and SAP provide powerful BPM tools for modeling business processes:

- IBM with the WebSphere Integration Developer
- SAP with the Process Composer, which is part of the NetWeaver Composition Environment

While the integration of a ruleset, which has been exposed as a service, is easily possible in the WID, the configuration steps in SAP's Administration Console make it more difficult to integrate the service in the Business Process. Nevertheless the configuration steps have to be done once and are often non-recurring.
The integration of SAP NW BRM with an application written in ABAP is very well possible using BRFplus' built in Expression Type called "BRMS Connector". The BRMS Connector provides a mechanism to delegate incoming calls from an ABAP application to an external BRMS, which in this case was SAP NW BRM. SAP NW BRM provides a built in JCO Server (JCO = Java Connector) in it's Java based application server to handle the connection to the ABAP based application server.

Evaluating the interoperability aspects between these systems, with focus on the integration of IBM's BRMS in SAP's system (BRMS and ecosystem). Also elaborating to which extent such an interoperability is feasible:

This task embodied the evaluation part of this thesis, providing a way of integrating JRules with SAP's ecosystem and their BRMS. Since both IBM and SAP offer products to

implement a Service Oriented Architecture (WebSphere and NetWeaver), the integration of a ruleset, which has been modeled in JRules and is exposed as a Web Service, should not be difficult. And indeed, the integration of JRules with a business process modeled in SAP is very similar to the integration of SAP NW BRM with the business process – there are only a few changes in the configuration steps of the administration console.
However, the direct integration of JRules with SAP's Java based application server is not supported by IBM and therefore not an option for interoperability.
My opinion towards the integration between JRules and SAP's BRMS has been the topic of chapter 7.6. Summarized, the interoperability between SAP NW BRM and JRules is not possible, due to a lack of necessary functionality in SAP NW BRM to enable this, while the interoperability between JRules and BRFplus is possible, because the BRMS Connector is a built in mechanism to model a Pass-Through Rule, which delegates incoming ABAP-calls to JRules.

Outlook:

As already mentioned in chapter 7.6, in the future the BRMS of IBM and SAP might be able to interoperate using the standards conducted by the organization W3C. The Rule Interchange Format is a promising approach for enabling the exchange of rules between different BRMS. With JRules it is already possible to export modeled rules and transform them to the RIF format, using the RIF-dialect for production rules, which has also been proven using tests. SAP still has to enhance it's BRMS with the functionality to export and import modeled rules and also has to expose the technical rule language, which the rule engine uses to execute the rules. But since SAP is enhancing it's BRMS with every release of the Composition Environment, this functionality can soon be included in SAP NW BRM, which then enables the evaluation of the interoperability between JRules and SAP NW BRM through RIF.
Another possible scenario in the future is that the BRMS start adhering to the standard rule language RuleML, which is also being developed and enhanced – version 1.0 has been published in 2010. If both JRules and SAP NW BRM comply to RuleML, it is possible to model rules in the language offered by RuleML and deploy it on both BRMS. Nevertheless the deployment alone is not sufficient, the semantics also have to be preserved, since deploying the same rules on both BRMS, which execute them differently, is not intended.
Considering BRFplus, it is being enhanced continuously, but the interoperability with other BRMS through the Import/Export-method is not possible right now, because rules

can't be exported. However, it is also not necessary, since the BRMS Connector enables seamless interoperability with other BRMS, which are based on Java.

9 Appendix

9.1 Code for the standalone JCO Server

```
public class MyJCoServer extends JCO.Server {
public MyJCoServer(String gwhost, String gwserv, String progid,
boolean isUnicode, IRepository repository) {
super(gwhost, gwserv, progid, repository);
this.setProperty("jco.server.unicode", isUnicode ? "1" : "0");
}
protected void handleRequest(JCO.Function function) {
JCO.ParameterList input = function.getImportParameterList();
JCO.ParameterList output = function.getExportParameterList();
JCO.ParameterList tables = function.getTableParameterList();
System.out.println("INPUT:");
System.out.println(input);
System.out.println("OUTPUT:");
System.out.println(output);
System.out.println("TABLES:");
System.out.println(tables);
// Extract the parameters from Table "ITS_CALL_PARAMETER"
Table paramsMap = input.getTable("ITS_CALL_PARAMETER");
System.out.println(paramsMap);
String project = "";
String ruleset = "";
if (paramsMap == null) {
try {
throw new J2EEAbapException("INVALID_INPUT", "Execution
params");
} catch (J2EEAbapException e) {
e.printStackTrace();
}
} else {
paramsMap.firstRow();
for (int i = 0; i < paramsMap.getNumRows(); i++) {
String paramKey = paramsMap.getString("NAME");
String paramValue = paramsMap.getString("VALUE");
System.out.println("Name: " + paramKey + " Value: "
+ paramValue);
```

```
if (paramKey.equalsIgnoreCase("project"))
project = paramValue;
else if (paramKey.equalsIgnoreCase("ruleset"))
ruleset = paramValue;
else
System.out.println("Unknown Parameter: " + paramKey + " = "
+ paramValue);
paramsMap.nextRow();
}
}
System.out.println("Rule execution parameters:");
System.out.println("project=" + project + ", ruleset=" + ruleset);
// Parse the incoming XML-File "IV_INPUT_XML"
System.out.println("handleRequest(" + function.getName() + ")");
try {
// Initiate DOM Parser
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
DocumentBuilder parser = dbf.newDocumentBuilder();
byte[] xmlBytes = input.getByteArray("IV_INPUT_XML");
if (xmlBytes != null) {
System.out.println("XML Byte Array Length: " + xmlBytes.length);
InputSource xmlFileInput = new InputSource(
new ByteArrayInputStream(xmlBytes));
// DOM
Document dom = parser.parse(xmlFileInput);
// set "ApprovalNeeded"-Tag
System.out.println("Fetching XML Content...");
Element root = dom.getDocumentElement();
String country = root.getElementsByTagName(
"FDTNS:REQUESTOR_COUNTRY").item(0).getTextContent();
String amountString = root.getElementsByTagName(
"FDTNS:TOTAL_AMOUNT").item(0).getTextContent();
Element approvalNode = (Element) root.getElementsByTagName(
"FDTNS:APPROVAL_NEEDED").item(0);
double amount = Double.parseDouble(amountString);
boolean result = invokeJRulesWebService(country, amount);
Text text;
if (result) {
text = dom.createTextNode("X");
approvalNode.appendChild(text);
}
// Print received XML to Console
printXML(dom);
byte[] outputValue = doc2bytes(dom);
output.setValue(outputValue, "EV_OUTPUT_XML");
} else {
throw new J2EEAbapException("INVALID_INPUT", "Context XML");
}
} catch (SAXException e) {
System.out.println("Problem creating XMLReader");
e.printStackTrace();
} catch (FileNotFoundException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} catch (J2EEAbapException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} catch (ParserConfigurationException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} catch (TransformerException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
/**
 * Print the XML to the Console
 *
 * @param doc
 * @throws TransformerException
 */
private void printXML(Document doc) throws TransformerException {
// set up a transformer
TransformerFactory transfac = TransformerFactory.newInstance();
Transformer trans = transfac.newTransformer();
trans.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION, "no");
trans.setOutputProperty(OutputKeys.INDENT, "yes");
```

```
// create string from xml tree
StringWriter sw = new StringWriter();
StreamResult result = new StreamResult(sw);
DOMSource source = new DOMSource(doc);
trans.transform(source, result);
String xmlString = sw.toString();
// print xml
System.out.println("XML Datei:\n\n" + xmlString);
}
/**
 * Method to transform a Document-Element into ByteArray
 *
 * @param node
 * @return
 */
public static byte[] doc2bytes(Node node) {
try {
Source source = new DOMSource(node);
ByteArrayOutputStream out = new ByteArrayOutputStream();
StringWriter stringWriter = new StringWriter();
Result result = new StreamResult(out);
TransformerFactory factory = TransformerFactory.newInstance();
Transformer transformer = factory.newTransformer();
transformer.transform(source, result);
return out.toByteArray();
} catch (TransformerConfigurationException e) {
e.printStackTrace();
} catch (TransformerException e) {
e.printStackTrace();
}
return null;
}
/**
 *
 * @param requestorCountry
 * @param totalAmount
 * @return approvalNeeded
 */
private boolean invokeJRulesWebService(String requestorCountry, double
totalAmount) {
```

```
//BigDecimal amount = new BigDecimal(totalAmount);
long amount = (long) totalAmount;
DecisionServiceApprovalRules_ServiceLocator locator = new
DecisionServiceApprovalRules_ServiceLocator();
try {
DecisionServiceApprovalRules_PortType portType =
locator.getDecisionServiceSOAP9152202211();
System.out.println("(JRules) Setting parameters...");
DecisionServiceRequest req = new DecisionServiceRequest();
DecisionServiceResponse res = new DecisionServiceResponse();
req.setDecisionID("12345");
req.setRequestorCountry(requestorCountry);
req.setTotalAmount(amount);
System.out.println("(JRules) Finished.");
System.out.println("(JRules) Invoking WebService...");
res = portType.executeDecisionService(req);
System.out.println("(JRules) Finished.");
String approvalNeeded = res.getApprovalNeeded();
System.out.println("(JRules) Result : " + approvalNeeded);
if (approvalNeeded.equalsIgnoreCase("X"))
return true;
else
return false;
} catch (ServiceException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} catch (DecisionServiceException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} catch (RemoteException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
return false;
}
}
```

9.2 Code for “XML Parser” Java Component in WID

```
public class XMLParserImpl1 {
/**
 * Default constructor.
 */
public XMLParserImpl1() {
super();
}
/**
 * Return a reference to the component service instance for this implementation
 * class.  This method should be used when passing this service to a partner
reference
 * or if you want to invoke this component service asynchronously.
 *
 * @generated (com.ibm.wbit.java)
 */
@SuppressWarnings("unused")
private Object getMyService() {
return (Object) ServiceManager.INSTANCE.locateService("self");
}
/**
 * This method is used to locate the service for the reference
 * named "JRulesInterfacePartner".  This will return an instance of
 * {@link JRulesInterface}.  If you would like to use this service
 * asynchronously then you will need to cast the result
 * to {@link JRulesInterfaceAsync}.
 *
 * @generated (com.ibm.wbit.java)
 *
 * @return JRulesInterface
 */
private JRulesInterface _JRulesInterfacePartner = null;
public JRulesInterface locateService_JRulesInterfacePartner() {
if (_JRulesInterfacePartner == null) {
_JRulesInterfacePartner = (JRulesInterface) ServiceManager.INSTANCE
.locateService("JRulesInterfacePartner");
}
return _JRulesInterfacePartner;
```

```
}
/**
 * Method generated to support implementation of operation "getRiulesData" defined for
WSDL port type
 * named "MyInterface".
 *
 * Please refer to the WSDL Definition for more information
 * on the type of input, output and fault(s).
 */
public byte[] getRulesData(byte[] iNPUTXML) {
// Initiate DOM Parser
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
DocumentBuilder parser;
try {
parser = dbf.newDocumentBuilder();
if (iNPUTXML != null) {
InputSource xmlFileInput = new InputSource(
new ByteArrayInputStream(iNPUTXML));
// InputSource xmlFileOutput = new InputSource(new
// ByteArrayInputStream(xmlBytesOutput));
// DOM
Document dom;
dom = parser.parse(xmlFileInput);
// set "ApprovalNeeded"-Tag
System.out.println("Fetching XML Content...");
Element root = dom.getDocumentElement();
String country = root.getElementsByTagName(
"FDTNS:REQUESTOR_COUNTRY").item(0).getTextCon
tent();
String amountString = root.getElementsByTagName(
"FDTNS:TOTAL_AMOUNT").item(0).getTextContent(
);
Element approvalNode = (Element) root.getElementsByTagName(
"FDTNS:APPROVAL_NEEDED").item(0);
double amount = Double.parseDouble(amountString);
//invoke the web service
String res = invokeWebService(country,
Double.toString(amount));
System.out.println(res);
if (res.equalsIgnoreCase("X")) {
Text text = dom.createTextNode("X");
approvalNode.appendChild(text);
}
printXML(dom);
byte[] outputValue = doc2bytes(dom);
return outputValue;
}
} catch (ParserConfigurationException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
catch (SAXException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} catch (TransformerException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
return new byte[0];
}
private String invokeWebService(String country, String amount) {
JRulesInterface test = locateService_JRulesInterfacePartner();
String result = test.InterfaceForJRulesWS(country, amount);
return result;
}
/**
 * Method generated to support the async implementation using callback
 * for the operation (@link
jco_server_mediationmodule.interfaceforjrulesws.JRulesInterface#InterfaceForJRulesWS(Str
ing part0, String part1))
 * of java interface (@link
jco_server_mediationmodule.interfaceforjrulesws.JRulesInterface)
 * @see
jco_server_mediationmodule.interfaceforjrulesws.JRulesInterface#InterfaceForJRulesWS(Str
ing part0, String part1)
 */
public void onInterfaceForJRulesWSResponse(Ticket __ticket,
String returnValue, Exception exception) {
//TODO Needs to be implemented.
}
```

```
/**
 * Method to transform a Document-Element into ByteArray
 *
 * @param node
 * @return
 */
public static byte[] doc2bytes(Node node) {
try {
Source source = new DOMSource(node);
ByteArrayOutputStream out = new ByteArrayOutputStream();
StringWriter stringWriter = new StringWriter();
Result result = new StreamResult(out);
TransformerFactory factory = TransformerFactory.newInstance();
Transformer transformer = factory.newTransformer();
transformer.transform(source, result);
return out.toByteArray();
} catch (TransformerConfigurationException e) {
e.printStackTrace();
} catch (TransformerException e) {
e.printStackTrace();
}
return null;
}
private void printXML(Document doc) throws TransformerException {
// set up a transformer
TransformerFactory transfac = TransformerFactory.newInstance();
Transformer trans = transfac.newTransformer();
trans.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION, "no");
trans.setOutputProperty(OutputKeys.INDENT, "yes");
// create string from xml tree
StringWriter sw = new StringWriter();
StreamResult result = new StreamResult(sw);
DOMSource source = new DOMSource(doc);
trans.transform(source, result);
String xmlString = sw.toString();
// print xml
System.out.println("XML Datei:\n\n" + xmlString);
}
}
```

9.3 ABAP code to call BRFplus

```
lo_factory = cl_fdt_factory => if_fdt_factory~get_instance(
lo_function = lo_factory->get_function( lv_function ).
lo_context = lo_function->get_process_context().
lo_context->set_value( iv_id = lv_object_id
    ia_value = lv_value).
lo_function->process( EXPORTING io_context = lo_context
  IMPORTING eo_result = lo_result ).
  lo_result->get_value( IMPORTING ea_value = lv_result).
```

Bibliography

[1] http://en.wikipedia.org/wiki/IBM

[2] http://en.wikipedia.org/wiki/SAP

[3] SOA for SAP Customers: Accelerating Customer Value with SOA. Lloyd Adams, July 2007

[4] SAP NetWeaver Composition Environment:Composite Applications ermöglichen flexible Geschäftsprozesse: Prof. Dr. Klaus Kruczynski, Robert Buschmann

[5] SAP Whitepaper: Interoperability of SAP NetWeaver and IBM WebSphere, 2004

[6] EAI to SOA, Evolution or Revolution. Bruce Twite, the marlo group, 2007

[7] Bruce Silver Associates: SAP NetWeaver BPM White Paper, 2009

[8] http://www.agileitarchitecture.com/2007/12/agile-business-rule-development.html

[9] IBM Websphere ILOG JRules Infocenter: http://publib.boulder.ibm.com/infocenter/brjrules/v7r1/

[10] SAP Whitepaper: Business Rules Framework plus – The Very Basics, Carsten Ziegler, Thomas Albrecht, 2008

[11] ILOG: Building Cutting Edge Applications with Business Rules Technology, Lionel Macé

http://www.nljug.org/pages/events/content/jspring_2006/sessions/00002/slides/

[12] http://en.wikipedia.org/wiki/Conflict_resolution_strategy

[13] http://en.wikipedia.org/wiki/Rete_algorithm

[14] SAP Whitepaper: Understanding the Rule Engine, 2008

[15] IBM Whitepaper: Exploring IBM WebSphere ILOG JRules Integration with IBM WebSphere Process Server, 2009

[16] http://en.wikipedia.org/wiki/Websphere

[17] IBM Redbook: WebSphere Application Server V7.0: Concepts, Planning, and Design

[18] IBM Whitepaper: IBM WebSphere Process Server for Multiplatforms

[19] IBM Education Assistant: http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0

[20] http://en.wikipedia.org/wiki/ILOG

[21] Overview of ILOG JRules and WebSphere Process Server integration: http://www.ibm.com/developerworks/websphere/library/techarticles/1002_duan/1002_duan.html

[22] http://en.wikipedia.org/wiki/SAP_Enterprise_Services_Architecture

[23] SAP Collaboration Workspace: https://cw.sdn.sap.com/cw/docs/DOC-106203

[24] SAP NetWeaver CE 7.2 Library:
http://help.sap.com/saphelp_nwce72/helpdata/en/44/d958673ef05f4de10000000a11466f/frameset.htm

[25] SAP NetWeaver 7.0 EHP1 Library:
http://help.sap.com/saphelp_nw70ehp1/helpdata/en/97/ca03460a6643429e80b5c45dff5bed/content.htm

[26] SAP Community Network Forums: http://forums.sdn.sap.com/thread.jspa?threadID=1800173&tstart=30

[27] SAP Collaboration Workspace: https://cw.sdn.sap.com/cw/docs/DOC-102733

[28] http://en.wikipedia.org/wiki/Semantic_web

[29] http://en.wikipedia.org/wiki/Rule_Interchange_Format

[30] Please Pass the Rules: A Rule Interchange Demonstration. Gary Hallmark, Christian de Sainte Marie, Marcos Didonet Del Fabro, Patrick Albert, and Adrian Paschke, 2008

[31] http://en.wikipedia.org/wiki/RuleML

[32] http://en.wikipedia.org/wiki/JSR_94

List of Tables

List of Figures