IT of the Future: Semantic Cloud Architecture

Preface

The series of articles “From Business as Usual to Knowledge-Driven Architecture” [1] (http://semanticweb.com/from-business-as-usual-to-knowledge-driven-architecture-part-i_b2124) outlined enterprise IT of the future with integrated software and knowledge engineering, further expanding on ideas originally described in the book “Integration-ready Architecture and Design” [2].

This article focuses on the transitioning process with very practical “baby steps”, which require minimum upfront investment. The emphasis is on collaborative work of business and enterprise architects with the Business Architecture Sandbox for Enterprise, the BASE, demonstrated at the Semantic Tech and Business Conference 2012.

The discussed approach is gradually shifting the focus of IT from technology to information by standardizing business event processing, placing the seeds of semantic technology in the current business ground, and establishing a self-sustaining process of transformation to semantic cloud architecture. The article provides the context and speaks technical details for this transition.

Table of Contents

Transitioning from IT as we know today to Semantic Cloud Architecture

Enterprise IT as we know today is slowly disappearing. Some companies are transitioning their IT to a cloud. But even a bigger transformation is on the way. The transformation from complex reality of multiple integrated systems to a much simpler semantic architecture, which is more focused on the information itself than on the systems supporting information.

Yes, Enterprise IT can be and will be very simple. And IT will come back to its original meaning, where Information comes first. More than 50% of IT budget is currently dedicated to managing technical systems, not managing information.

Why? – Different types of information were historically present by different systems.

Semantic Technologies will change this forever by offering a unified landscape for all types of information.

Wait a minute! *Specific* data tables in *specific* applications make *specific* SQL queries perform faster. True! But in the increasingly interconnected business, integration efforts outweigh the benefits of specific approaches to specific data. Plus, a growing art and science of Big Data helps us absorbing a full story, not just small pieces of the story.

Just imagine that all enterprise information is collected in a single component, a cloud of Linked Data. Another component is Conversational Semantic Decision Support, a flexible mechanism, which can handle information for us and keep behind the scene the technical pieces, like SPARQL, a logical query language, similar but simpler than SQL.

Collecting a complete set of enterprise information about business events, processes, and their relationships is a very challenging task. About 80% of this information is “tribal knowledge”, not captured properly or not captured at all. People constantly fill in the informational gaps with multiple meetings and phone calls. This is our “usual business” routine.

Computer programs have even less tolerance to informational gaps.
So, we need to *allow them (computer programs) to call us (subject matter experts).*
In other words we need to establish conversational approach to the process of collecting data and resolving uncertainties. There are two key figures in this process: a SME who can answer the questions related to missing information, and a system architect who can structure right questions by modeling a domain of the business events and processes.

While working with both groups in business and IT, I often had a pushback. “You want us to fly, but we are still learning to walk”. Transitioning from multiple systems into semantic enterprise architecture is an extra process, which is a hard sell in the current economic stage, when a lot of companies are struggling “just to keep lights on”.

To engage SMEs and architects in the process, they must get an easy entrance in the semantic world with immediate benefits, which would grow with every step in the semantic direction. Business Architecture Sandbox for Enterprise, the BASE, offers such an easy entrance and a platform to collaborate with IT on new approaches, while transitioning to Semantic Enterprise Architecture.

Business Architecture Sandbox for Enterprise (BASE)

Semantic Enterprise Architecture (SEA) can grow from the enterprise environment with well-established Service-Oriented Architecture. The BASE is instrumental in creating such environment and providing a natural transition to Semantic SOA with its beautiful simplicity. Simplified and standardized infrastructure makes a cloud solution even more attractive, significantly decreasing maintenance and development expenses.

Distributed infrastructure and the art of Big Data processing are changing the way we view and analyze information. Now, we have access to a full picture of the world of our interest. This is very different from the current approach where multiple applications deal with their pieces of the puzzle and deliver intermediate results to subject matter experts for further (mostly manual) integration.

The main goals and features of the Business Architecture Sandbox for Enterprise (BASE) are:

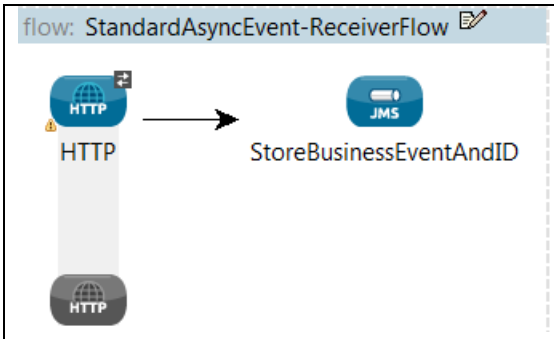
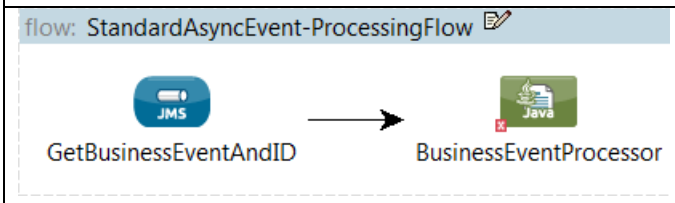
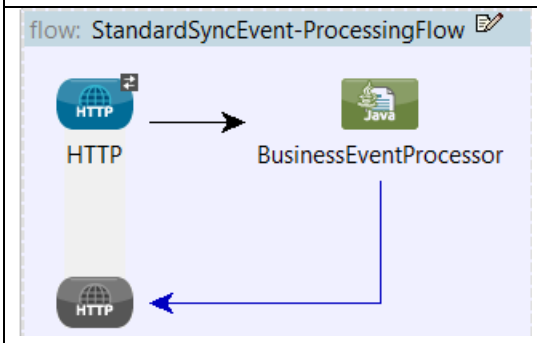
- Leverage SOA and standard event processing with high availability and fail-over features
- Create initial semantic models of business events and processes with their supporting components to improve development precision and prevent data and function duplication
- Provide semantic support for development and test to conduct these activities within the model before placing new components to production.
- Establish a playground for creating business workflow and services
  - Decrease the number of manual operations required for business changes
  - Reduce the opportunities for human errors and production problems
  - Standardize Restful API for multiple systems and 3-rd party developers
- Add a semantic layer to Enterprise Service Bus to enable semantic listening and prepare for canonical model integration with the systems speaking different business dialects
- The bottom line is to place the seeds of semantic technology in the current business ground and establish a self-sustaining process of transformation IT to semantic cloud architecture.

Standard event processing with the BASE, ESB/Mule, and ActiveMQ

The BASE is a portlet, which runs in the open source portal, Liferay [3]. The BASE is integrated with Mule, ESB [4], and Apache ActiveMQ [5]. This integrated system is configured as a cluster with multiple servers, providing a high availability and failover solution.

This basic SOA standardization provides the ground for service orchestration, reduces tight coupling of applications, and decreases production problems and maintenance efforts.

The BASE is set up as a standard platform for synchronous and asynchronous processing of any business events with the following message flows:

	**Standard Asynchronous Event Receiver Flow** 1. Receive an HTTP Restful Call 2. Store the Source and Parameters of the event in a proper message queue, each with its own priority. There are several prioritized queues according to Service Level Agreement.
	**Standard Asynchronous Event Processing Flow** 1. Get Business Event ID and Parameters 2. Instantiate Business Event Processor and orchestrate event processing. 3. Check Success in N re-trials. 4. Report success or execute a plan "B" (another Business Event Processor)
	**Standard Synchronous Event Processing Flow** 1. Receive HTTP Restful Call 2. Instantiate Business Event Processor and orchestrate event processing. 3. Send the resulting HTTP Response message

The BASE receives all business events as HTTP/HTTPS Restful requests for asynchronous or synchronous processing.
In the case of the asynchronous processing, each incoming request is stored in a message queue in the **Standard Asynchronous Event Receiver Flow** for execution in the **Standard Asynchronous Event Processing Flow**.
In the synchronous case, the **Standard Synchronous Event Processing Flow** provides the processing of the event resulting in the HTTP/HTTPS response.

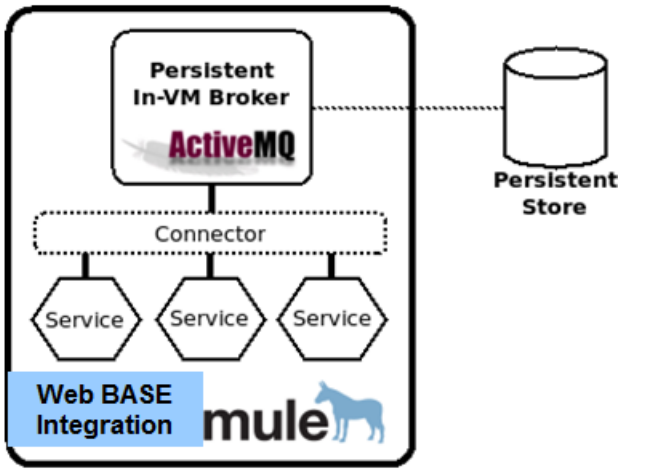
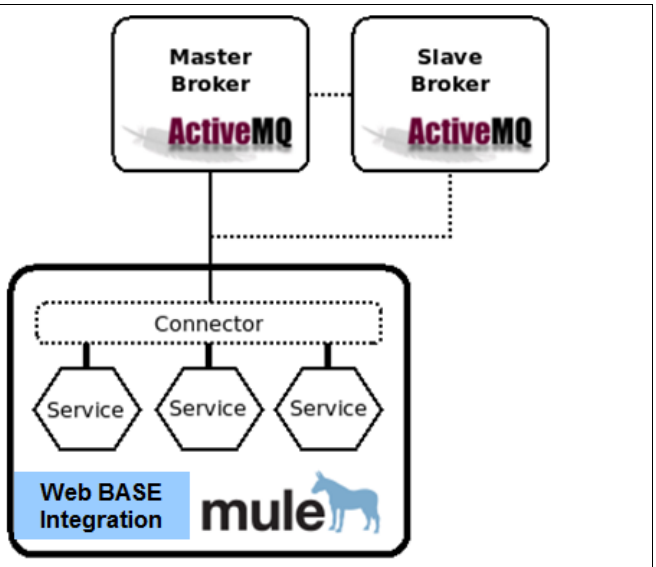
The messaging approach provides standard processing of any business event with necessary prioritization according to Service Level Agreements (SLA).
Two lines of the code below illustrate instantiation of the Business Event Processor based on the event name and processing the event with a specific parameter, like event ID.

```
WorkflowProcessor wfp = (WorkflowProcessor) Class.forName(businessEvent).newInstance();
wfp.run(eventID); // the "run" method starts a set of workflow states to process data
```

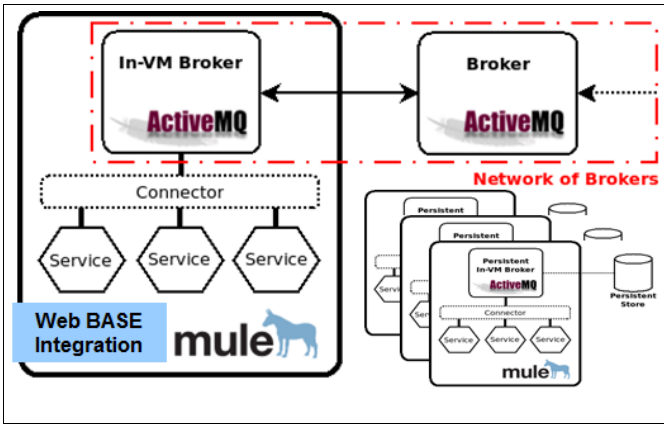
Note, that naming conventions play an essential role in reducing translation layers in the development and analysis. Semantically-rich environment starts with the catalog of business components and services – pure SOA products – and grows into Semantic SOA model, which establishes a business language, provides important dependency information, and allows us describing application behavior.

Integration strategy and Cluster Topology for high availability and fail-over

Mule Enterprise 3.2 and higher versions provide a standard high availability solution via Mule clustering with the internal data grid. Mule's Cluster can run multiple servers in the active-active mode and support multiple applications. JMS persistence is provided by the ActiveMQ (open source).

<table>
<tr>
<td> 
</td>
<td> 
</td>
</tr>
<tr>
<td>In the topology above, ActiveMQ broker has been configured to enable persistence of the JMS messages. It is good for scenarios with a single Mule instance, or where each instance and the JMS messages it processes can be functionally isolated.

This topology was used for a single Mule instance that uses JMS queues internally for reliably exchanging messages between its services. In the event of a crash, all messages pending delivery will have to wait until Mule, and its embedded ActiveMQ, has been restarted in order to be processed.</td>
<td>In the traditional Master-Slave topology communications between the Mule node and the ActiveMQ brokers happen over the wire, usually by using ActiveMQ's TCP transport. Consequently, this will lower overall performance. Moreover, it is necessary to configure Mule to handle the case when connecting to a remote broker isn't possible.

The Master-Slave topology is very common in production because of the high availability gained by deploying ActiveMQ as a pair of master and slave brokers. It is also a standard practice to have JMS providers deployed and operated in a centralized manner in corporate environments.</td>
</tr>
<tr>
<td> 
</td>
<td>The BASE integrated environment uses a combination of the JDBC Master-Slave topology with the Network of Brokers topology. This advanced solution provides Mule with distributed JMS queues and topics.

By co-locating Mule and the ActiveMQ broker within the same JVM, the network of brokers is accessed through the convenience of in-memory access.</td>
</tr>
</table>

A cluster can include two and more servers, where Mule and ActiveMQ are integrated with the BASE to create a very powerful trio for standard synchronous and asynchronous processing strategies.

Configuration lines below connect a master/slave pair of remote ActiveMQ brokers and uses the asynchronous retry policy provided with Mule ESB Enterprise:

```
<jms:activemq-connector name="JmsConnector"
                        brokerURL="failover:(${masterBrokerUrl},${slaveBrokerUrl})"
                        specification="1.1">

    <ee:retry-simple-policy frequency="3000"
                            asynchronous="true" />

</jms:activemq-connector>
```

URL to the Mule Management Console: ***http://server:8585/mmc***
URL to the Active MQ monitoring: ***http://server:8161/admin***
URL to all monitoring facilities integrated in the BASE:
http://server:8080/BASE-portlet/Lookup?appName=BASE&action-page=troubleshooting

Starting the URL with http://javaschool.com, you can see an example of a semantic model for a financial industry's company.

How semantic approach improves development and prevents duplications

Current tools and development practices often assume that developers have correct models of enterprise systems and relationships in their heads. This assumption is not true. Their assumed models are mismatched. This leads to data and function duplications, unnecessary system complexity, growing maintenance efforts, and production problems.

About 80% of this information is “tribal knowledge”, not captured properly or not captured at all. Business analysts and developers constantly fill in the informational gaps with multiple meetings and phone calls.

The Business Architecture Sandbox for Enterprise (BASE), a portlet in the Liferay portal environment, has some pre-defined skeleton-models and helps to create new semantic models on-the-fly.

The BASE splits this process in two steps:

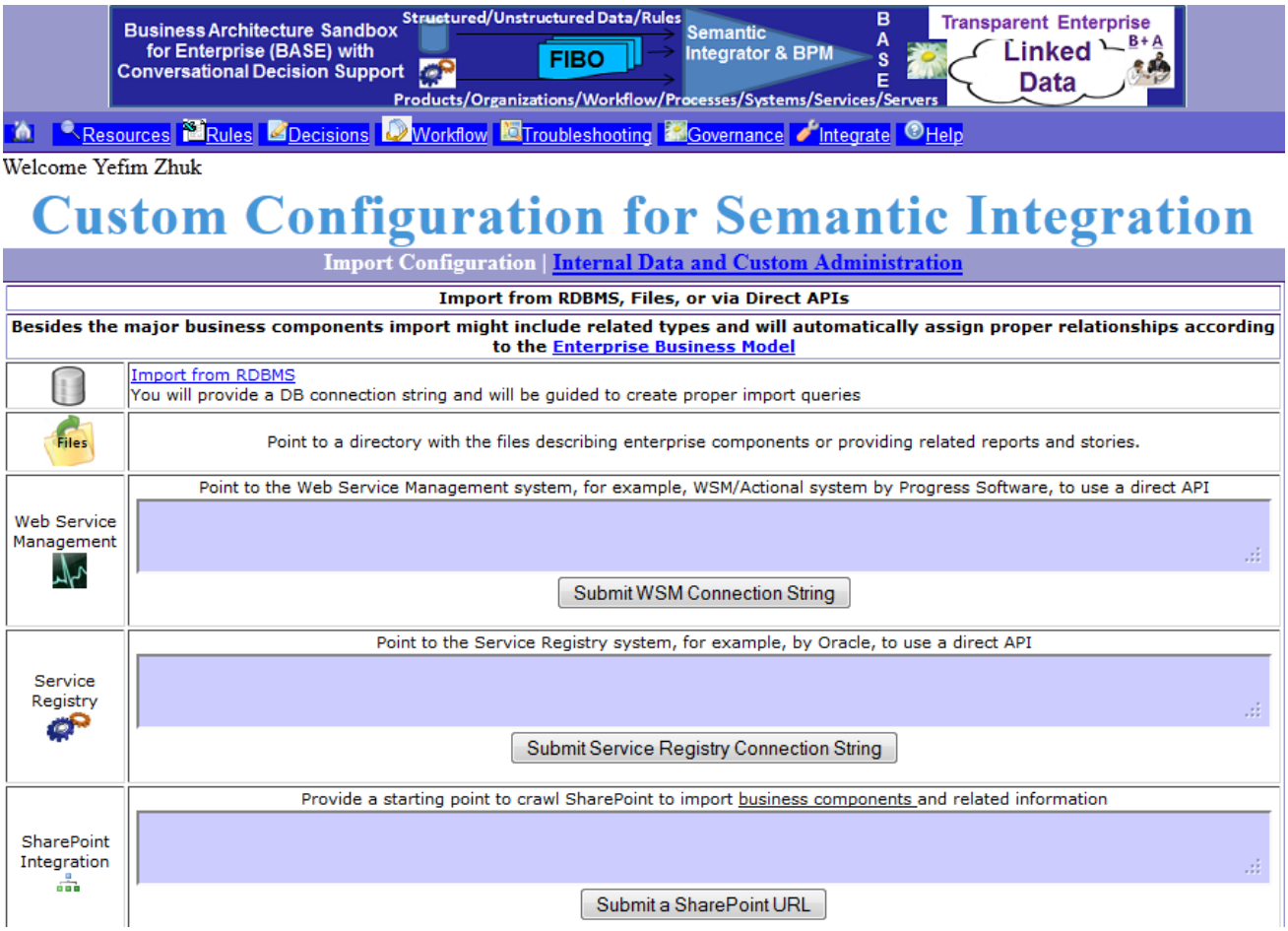
a) Creating a skeleton-model of object types with their potential relationships and

b) Filling in the skeleton-model with specific object instances and specific relationships. In each step, the BASE uses its conversational approach to define semantic models for a company, business process, or an event.

For example, such a model can describe enterprise business processes and supporting resources from IT perspectives. The skeleton-model will include such object types as Company and Products with supporting Business Processes, which in their turn will be implemented with Systems, Applications, and Services. All the above components will use Data Objects and Data Attributes. This simple skeleton will be later filled in with specific object instances with their names, descriptions, and relationships.

The screen above shows the beginning of the conversation that helps to fill in the skeleton-model on the left, just to initiate the semantic modeling process.

After the initial rounds of the scripted dialog, the system learns about the major business and system components of the enterprise. Now, the system is ready to searching over enterprise structured and unstructured data sources for more details and related smaller components. To satisfy this curiosity, we connect the BASE with the data sources via the Semantic Integration option.



The Semantic Integrator will scan the data looking for the names currently present in the initial model. Then, the integrator will extract related information to look for a vocabulary that will match a standard ontology. So far, Financial Industry Business Ontology (FIBO) plays this role in the BASE. The intention is to map enterprise specifics found in the data sources and build a specific semantic model of the enterprise based on a generic, standard model.

The integrator is not smart enough to accomplish this mapping task, but provides a significant help to a subject matter expert by arranging a semi-automated process.

Physicaldata

FIELDNAME	BUSINESSATTRIBUTENAME
COLUMNNAME	Column Name
CUSTOMMESSAGE	Custom Message
DATETIMESTAMP	Date Time Stamp
SQLERRORMESSAGE	SQL Error Message
TABLENAME	Table Name
ACHACCOUNTDESCRIPTION	Auto Clearing House Account Description
EID	Entity ID
ACCOUNTNUMBER	Account Number
ADJAMOUNT	Adjustment Amount
BATCHCYCLEDATE	Batch Cycle Date

The best results are received in mapping proprietary data field names to the Data Dictionary of FIBO.

After receiving DB connection hints, the integrator reads proprietary data structure from the tables of INFORMATION_SCHEMA. Usually, about 70% of unreadable data field names can be automatically mapped to their meaningful values from FIBO.

The integrator can understand a lot of abbreviations and can split proprietary Field Names in the left column into their meaningful Business Attribute Names in the right column. The resulting map represents a specific data dictionary of an enterprise based on a generic vocabulary in a standard ontology.

The main function of the Semantic Integrator is to create an initial Semantic model, while taking over the most boring and time consuming parts of data analysis. The BASE provides multiple options for business architects to update and grow the model.

The model represents a graph, which connects components with their relationships. The graph information is usually stored in a Triple Store [6], although it can be stored in a regular relational database. With a very simple and unified approach, you can describe enterprise components with their relationships. You can also capture any other content, like reports or connected stories, with multiple types of associated elements. The example of the data structure below will result in a semantic graph, which will link together related elements of multiple types.

ID	Type	Name	Description
01	Industry	Financial	Financial Services
02	Company	KeyBank	Banking Services
03	Company	Visa	Plastic Services

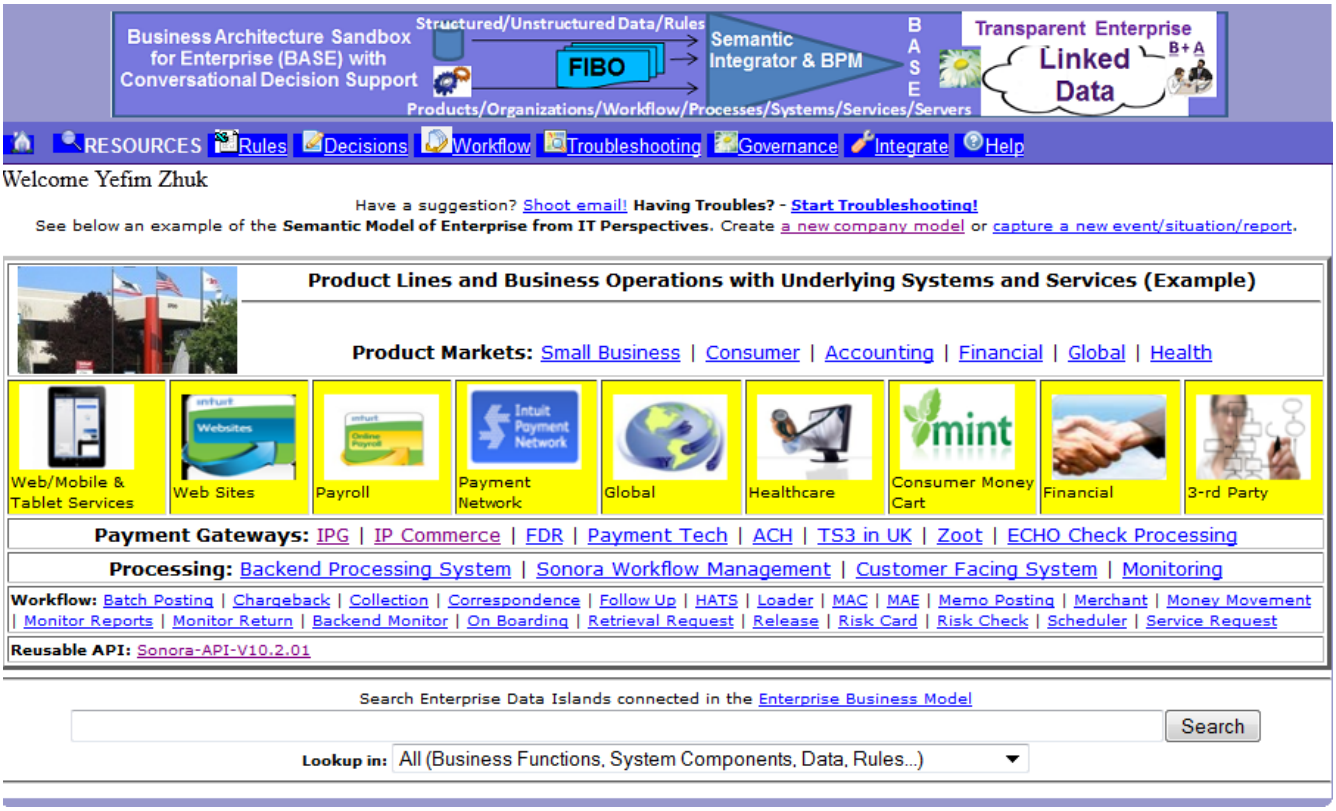
ID	Relationships	Related ID
02	Belongs To	01
03	Belongs To	01
02	Uses	03

This semantic model provides a true reflection of enterprise resources with their dependencies and enables model-driven development and testing, where a new component is created not in a vacuum but within an existing model.

The semantic model can be understood by a computer. This understanding enables new opportunities for collaborative work of subject matter experts (SME) and computer programs in business development and transferring “tribal knowledge” into decision making systems.

Managing Enterprise Resources with the Semantic SOA Model

The semantic SOA model turns enterprise data islands into linked and living knowledge.



The screen shot above provides an example of the top-level business and system components in a financial industry's company. The top-level components are linked to other components in the vertical and horizontal dimensions, effectively creating a semantic model of enterprise.

The screen shot of the Enrollment Line of Business provides its enterprise dependencies, which generally speaking can go beyond the vertical relationships presented on the screen.
The same physical components often have different names in different companies or even different

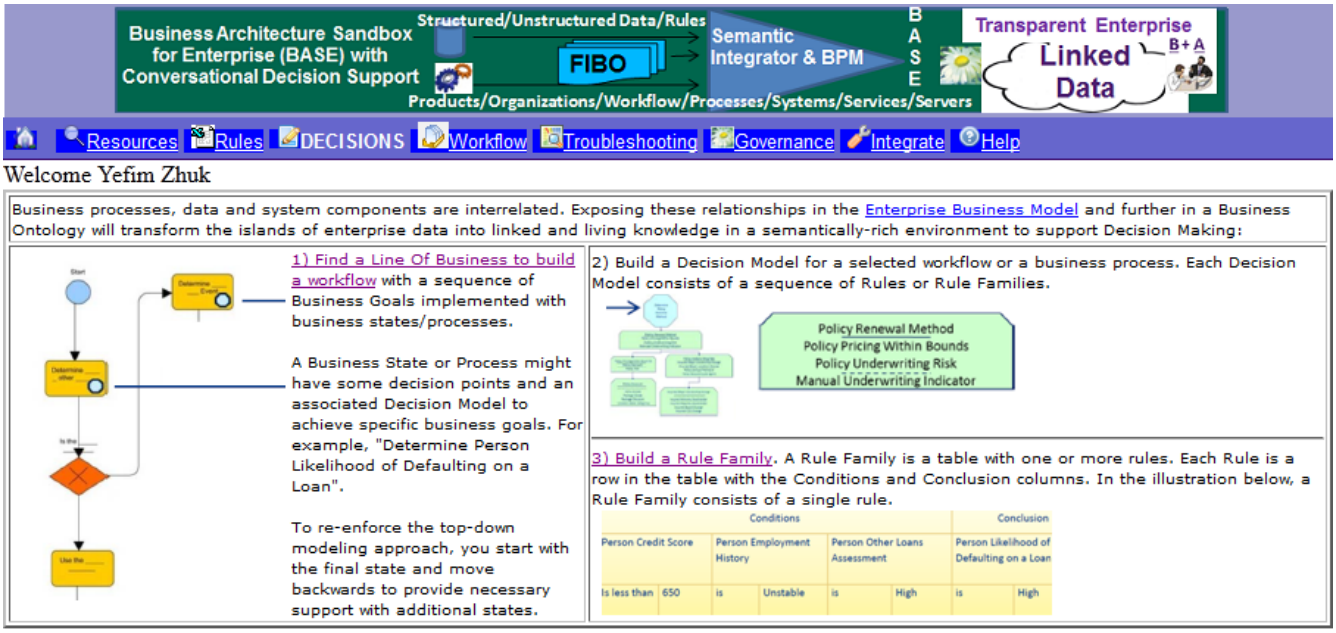
departments of the same company. The semantic mapping helps to resolve these differences and provides powerful support in search and decision making processes.

There are multiple industry tools, like Configuration Management Database (CMDB), to collect enterprise information. The BASE does not compete with these tools, but focuses on the semantic approach. The BASE combines the semantic approach with the development playground and allows architects and business analysts to collaborate on development tasks, while exercising and naturally expanding the semantic model.

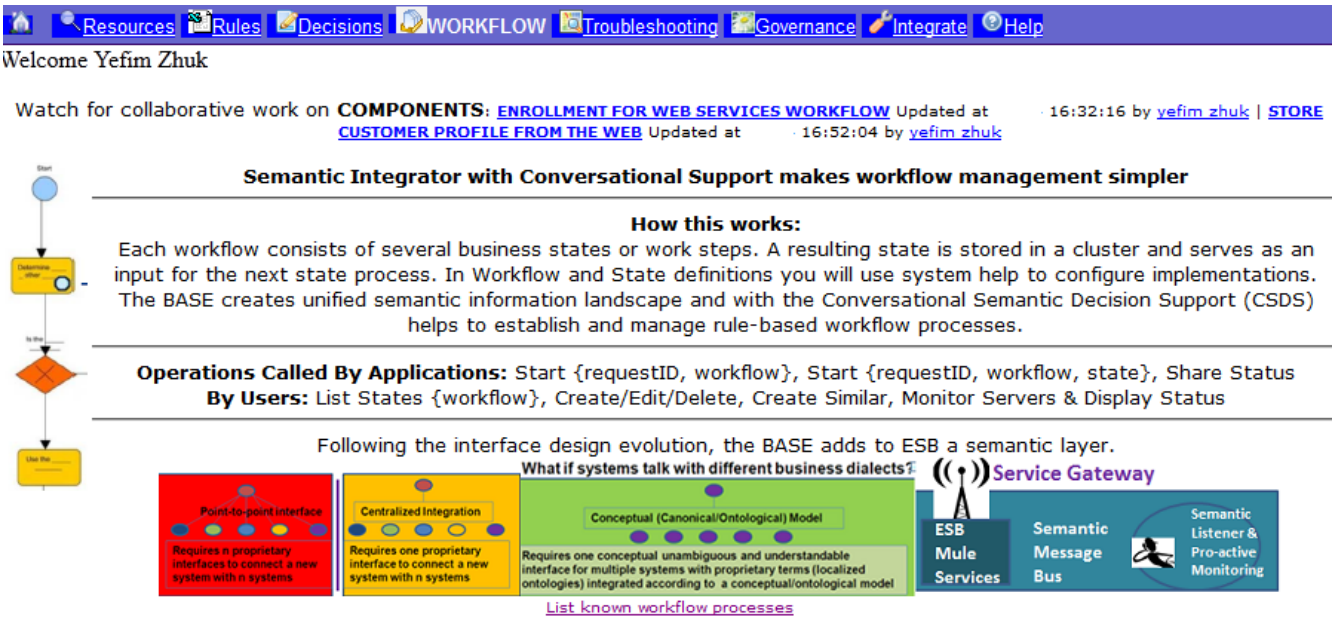
Development of Workflow Components within the semantic model

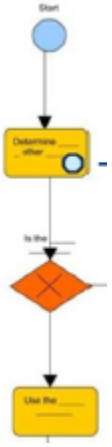
The BASE allows business analysts and developers for collaborative development of new components within existing semantic SOA model.

For example, a business analyst can type in the search text box: “build a new enrollment workflow” and the program will display existing workflows and components related to the Enrollment business line. The program will offer to check if anything in the existing model can fit the needs or be reused. Then, the BASE will start a conversational wizard helping to connect a new component to the existing model and will continue with the development recommendations.



Each workflow consists of several business states or work steps. A resulting state is stored in a cluster and serves as an input for the next state process. In Workflow and State definitions you will use system help to configure implementations.
The BASE creates a unified semantic information landscape and with the Conversational Semantic Decision Support helps to establish and manage rule-based workflow processes.

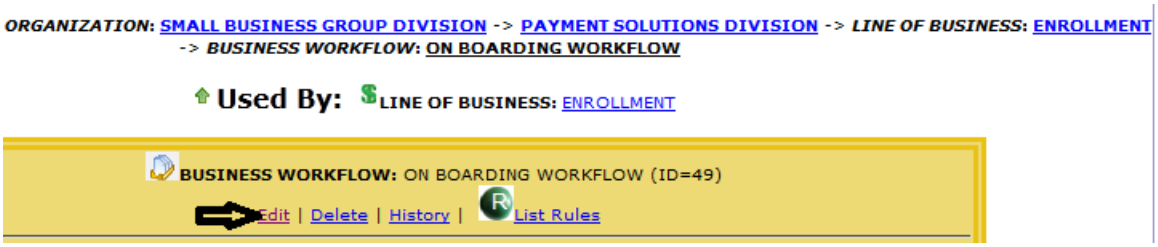




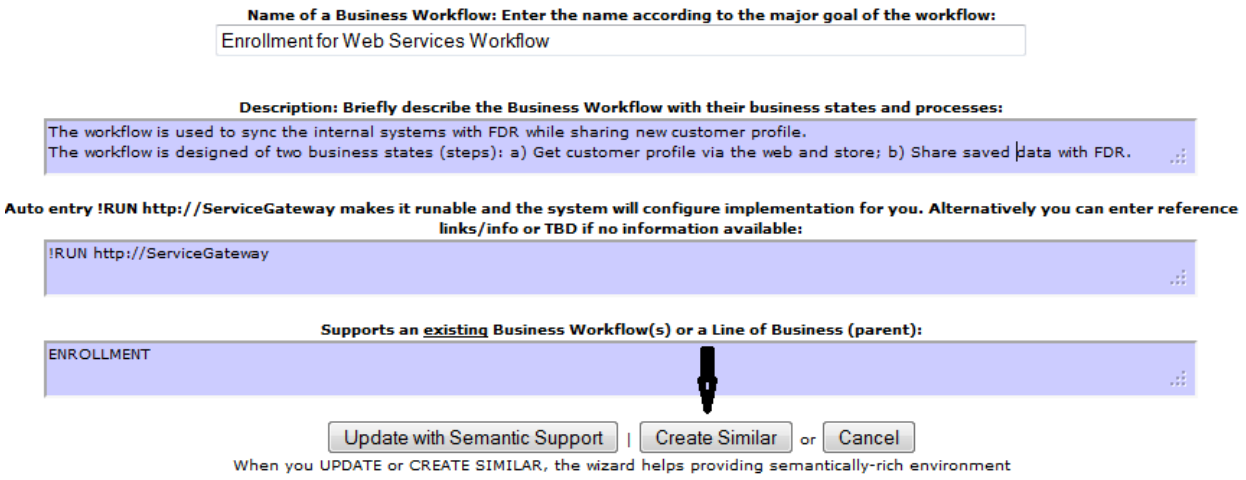
1) Build a workflow with a sequence of Business Goals implemented with business states/processes. A Business State or Process might have some decision points and an associated Decision Model to achieve specific business goals. For example, "Determine Person Likelihood of Defaulting on a Loan". To re-enforce the top-down modeling approach, you start with the final state and move backwards to provide necessary support with additional states.

2) Build a Decision Model for a selected workflow or a business process. Each Decision Model consists of a sequence of Rules or Rule Families.

The BASE offers "the lazy" development option – Create Similar. Creating a similar component by customizing some features is much easier than starting from scratch.

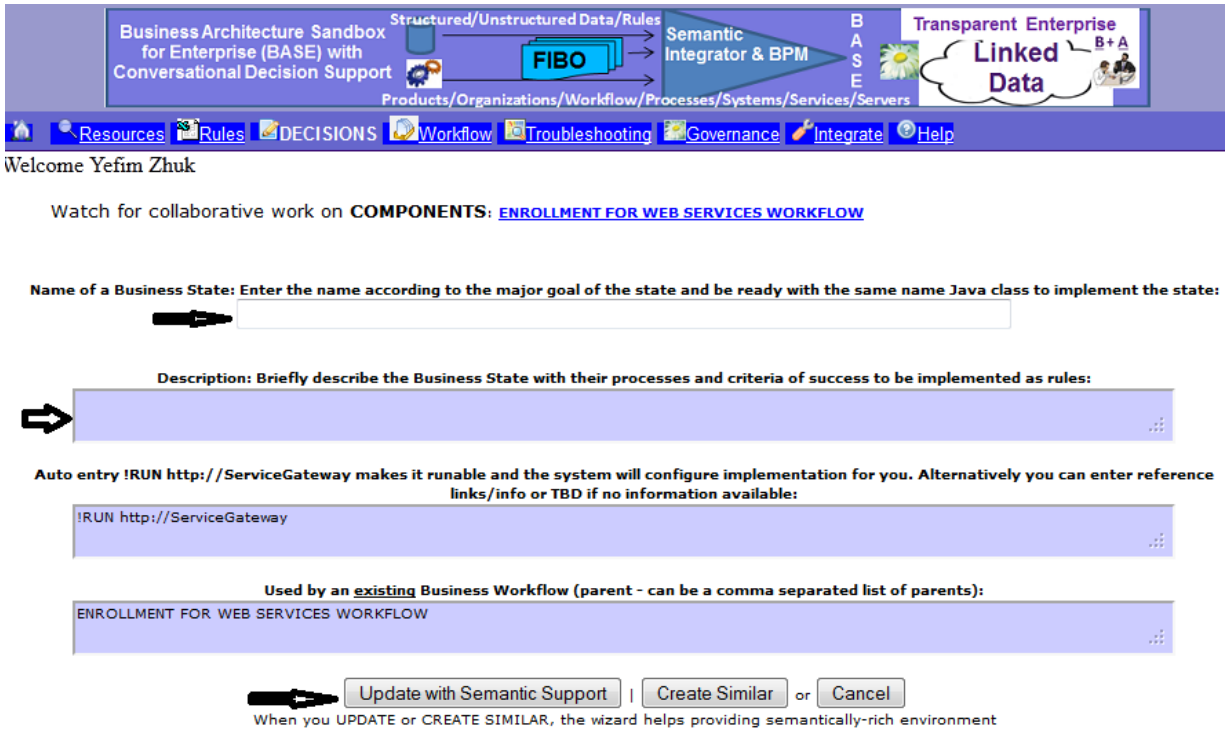


We'll pick up an existing component, in this case "On Boarding Workflow" and will use the EDIT control to customize this component according to our design.



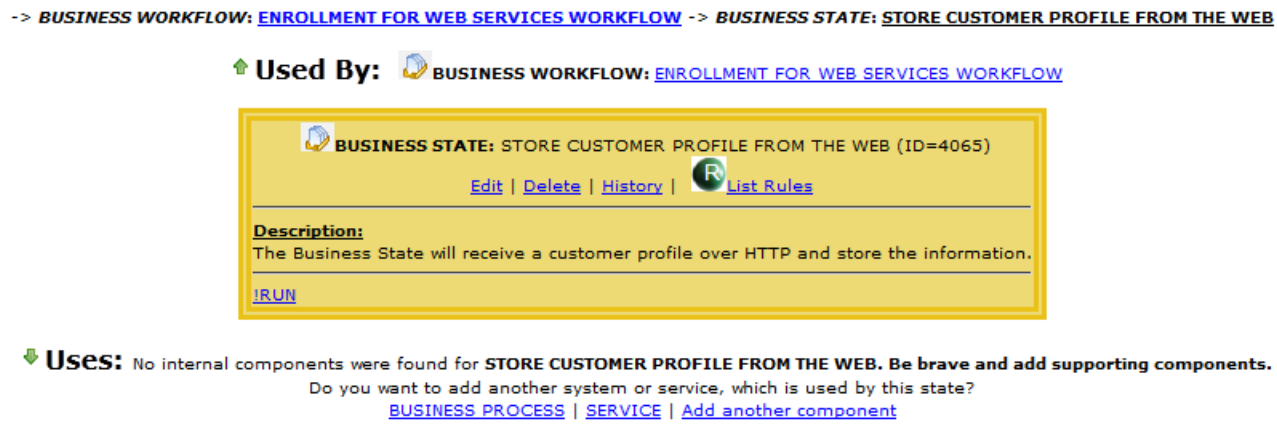
We will use the Create Similar control to end up with the new component, Enrollment for Web Services Workflow.

At this point we can add Business States to the workflow.

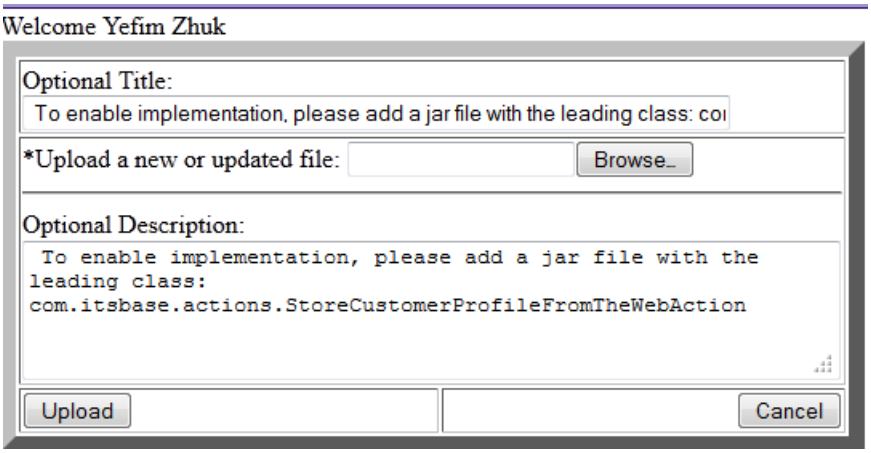


We'll provide the name and description for the Business State and will use the control "Update with Semantic Support". Before updating the business state, the program will check for unique and meaningful names, provide automatic linkage to the existing enterprise components, and make all changes visible to collaborative communities.

Each Business State or any other system component might have the "!RUN" - link, which allows developers to configure and then test these components within the model.



The "!RUN" link will prompt a developer to configure implementation of the Business State with the following message:
Implementation is not ready. Do you like to Cancel or Configure Implementation? To enable implementation, please add a jar file with the leading class: com.itsbase.actions.StoreCustomerProfileFromTheWebAction.



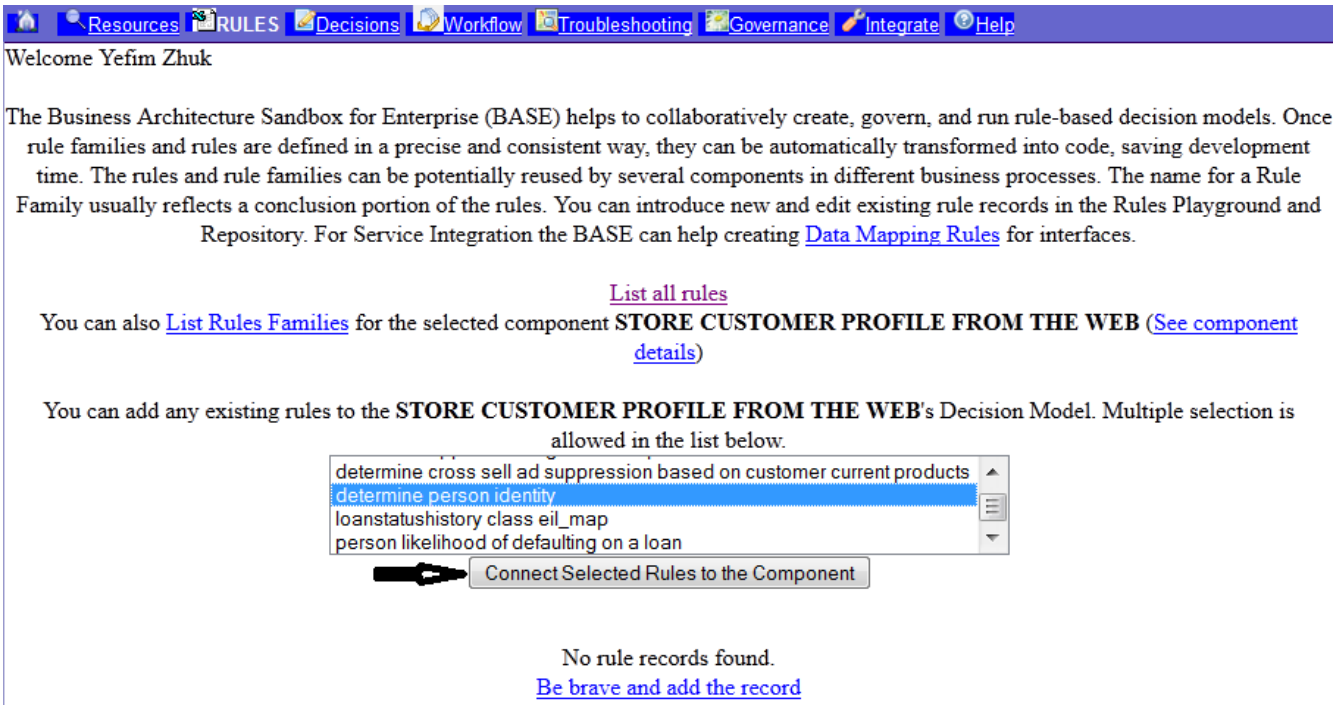
Note, that implementation of the Business State follows a simple naming convention, automatically expanding the semantic model with a new meaningful component name.

Establish the rules of the game with the Decision Tables

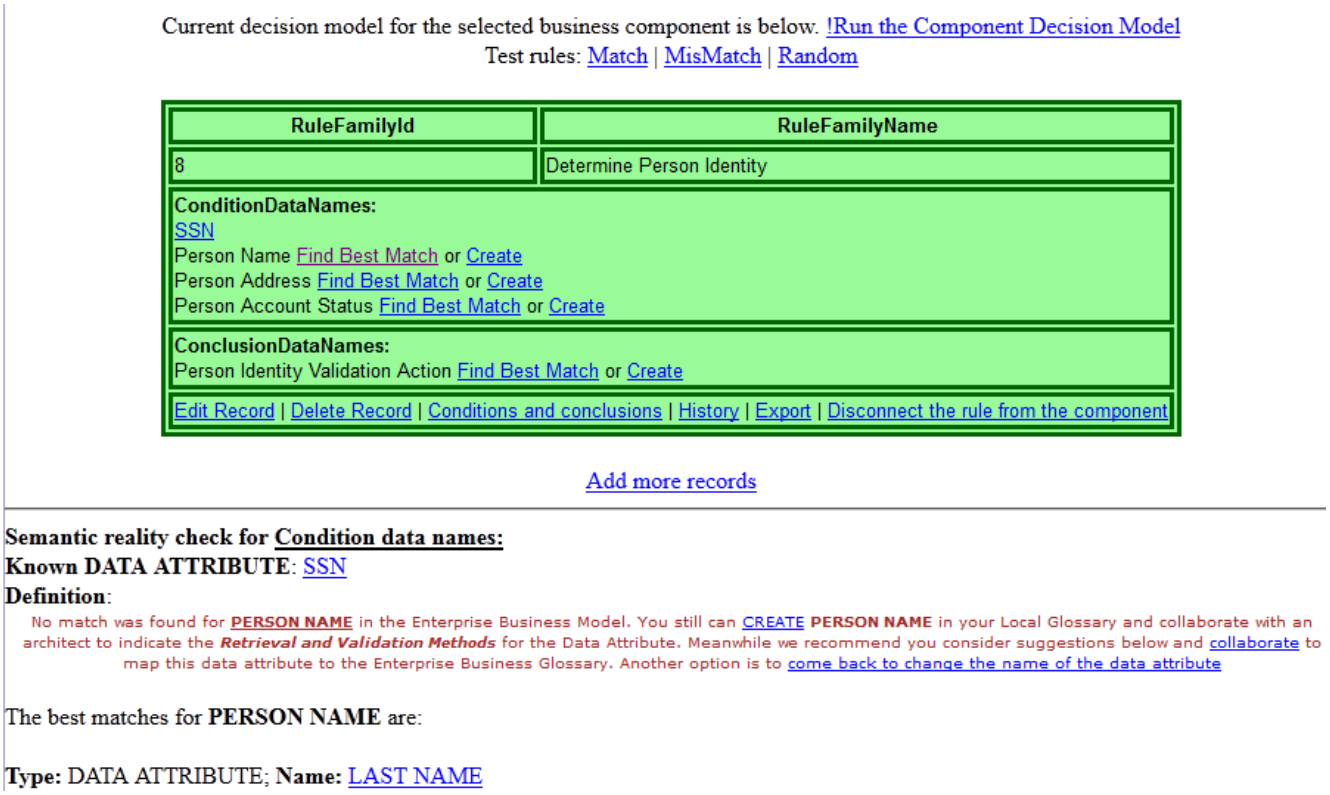
Each Business State usually includes some business logic. Built-in the BASE rules engine combined with the semantic model simplifies reuse of the rules and allows business analysts to directly participate in the development.

For example, the Business State "Store Customer Profile from the Web" might need to determine person's identity in the case when input data partially duplicate an existing profile. For example, if incoming name and social number are the same as in the existing profile but the address is different, the

question is which address or generally speaking which person's identity is the right one. This is a very common task, which can be potentially reused by multiple applications. The BASE makes this reuse easy and intuitive by sharing the rules and related data across applications.



We can create a new rule or reuse an existing rule by connecting the rule to a component. In this case we'll connect the rule "determine person identity" to the component **"STORE CUSTOMER PROFILE FROM THE WEB"**. The resulting screen is below.



The resulting screen displays this rule family as part of the component's Decision Model and

automatically produces the links for running and testing the model. The program provides the semantic reality check for Condition Data Names. Some data attributes, like SSN, are already in the system, and some are not. The program provides recommendations on mapping the data names to similar data attributes, existing in the system, or creating new attributes on-the-fly.

Let's take a look at this example. In the "Determine Person Identity" rule family, the rules are present as the rows and columns in the **decision table**. Each row is a separate rule, which considers several **conditions** for the following data attributes: SSN, Person Name, Person Address, and Person Account Status. Each row ends up with a **conclusion** based on the conditions.

In this example, the rules (rows) will check if a new profile duplicates any existing profile. If the name and SSN, received from the web, will match these values in an existing profile, but the address is not, then the program will look for a Person Account Status to decide if the existing profile is valid.

List all rules | List all rule-based components | Selected Rule Family is Determine Person Identity (id#8)
The rule family is used by the following components: ENROLLMENT FOR WEB SERVICES WORKFLOW | STORE CUSTOMER PROFILE FROM THE WEB

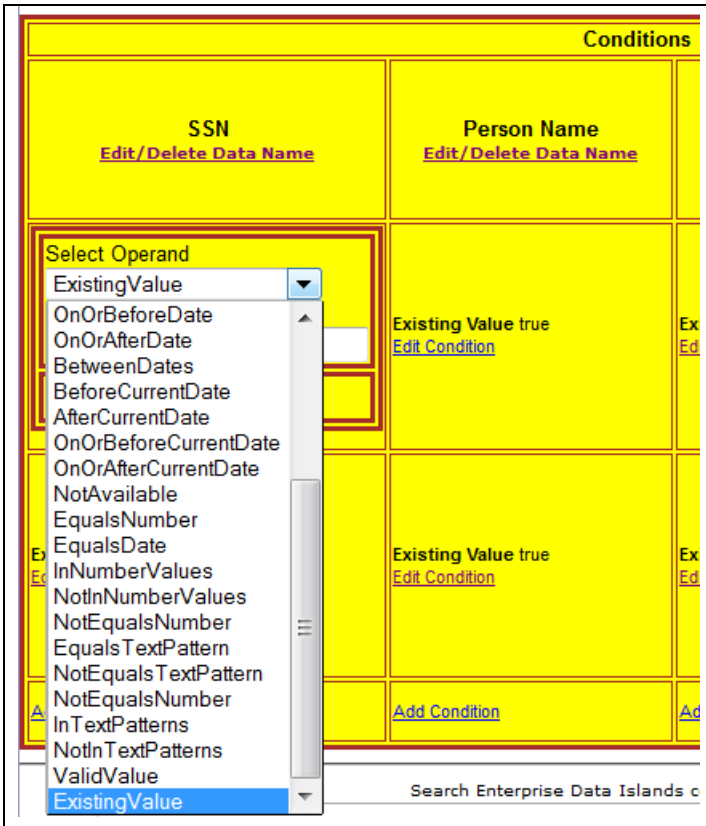
Conditions				Conclusions	
SSN Edit/Delete Data Name	**Person Name** Edit/Delete Data Name	**Person Address** Edit/Delete Data Name	**Person Account Status** Edit/Delete Data Name	**Person Identity Validation Action** Edit/Delete Data Name	
Existing Value true Edit Condition	**Existing Value** true Edit Condition	**Existing Value** false Edit Condition	**Valid Value** In Good Standing Edit Condition	Message: Existing account is valid for practical purposes Edit Conclusion	Delete Rule
Existing Value true Edit Condition	**Existing Value** true Edit Condition	**Existing Value** false Edit Condition	**Valid Value** Not Valid Edit Condition	Message: accept new customer profile instead of exisisting one. Edit Conclusion	Delete Rule
Add Condition	Add Condition	Add Condition	Add Condition	Add Conclusion	

Of course, a real set of rules is more complicated than that. The real rule family would include more rules/rows. Each rule in the decision table provides a situational description with the conditions and a conclusion, or even multiple conclusions. In the BASE, business analyst can specify a conclusion as an action, like it is done in the example below, or as a new data value. The decision table can use multiple conclusions and serve as a data transformation table, which transforms one data set into another data set. This data transformation is a very common task in the world of system integration.

“Data know how to handle data”

The most common problem with the rules engine is handling data within the rules. For example, JBoss Drools requires a developer to do massive data drilling with Java code before any rules can actually apply. Generally speaking, in our current environment data handlers belong to applications. For example, multiple applications where modified when US Government has changed SSN valid ranges.

The BASE uses semantic approach to shift this paradigm from “applications know how to handle data” to “data know how to handle data’. Each data attribute can be considered as an extended Java Bean, a placeholder for retrieval and data handling methods. In this world of linked data any application or a rule, which uses a data attribute, will automatically know how about major data handlers, because “data know how to handle data”.



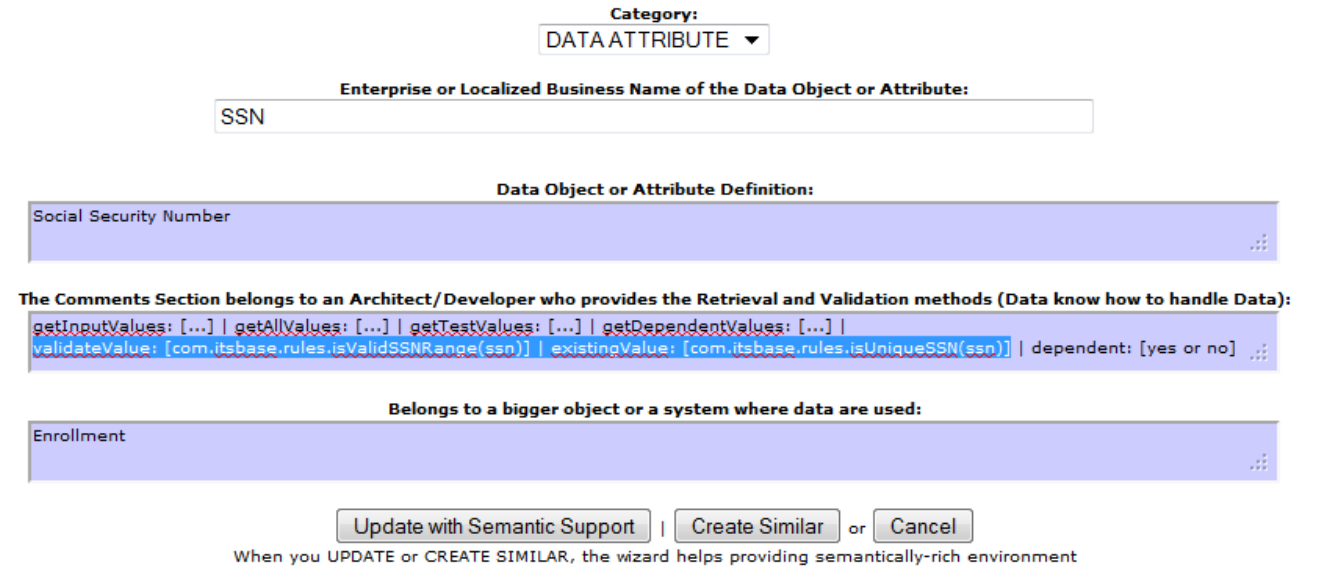
When “data know how to handle data”, any rule can use such powerful data conditions as “Valid Value”, “Existing Value”, and more.

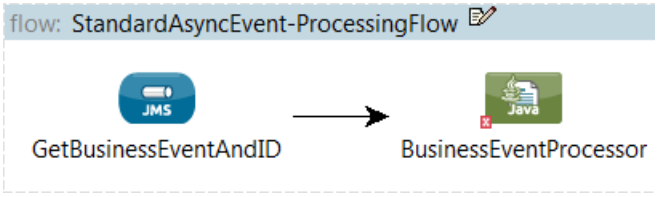
The screen shot on the left displays a subset of the long list of the conditional operands.

These conditions and data handlers are automatically available to all applications and rules that use the data.

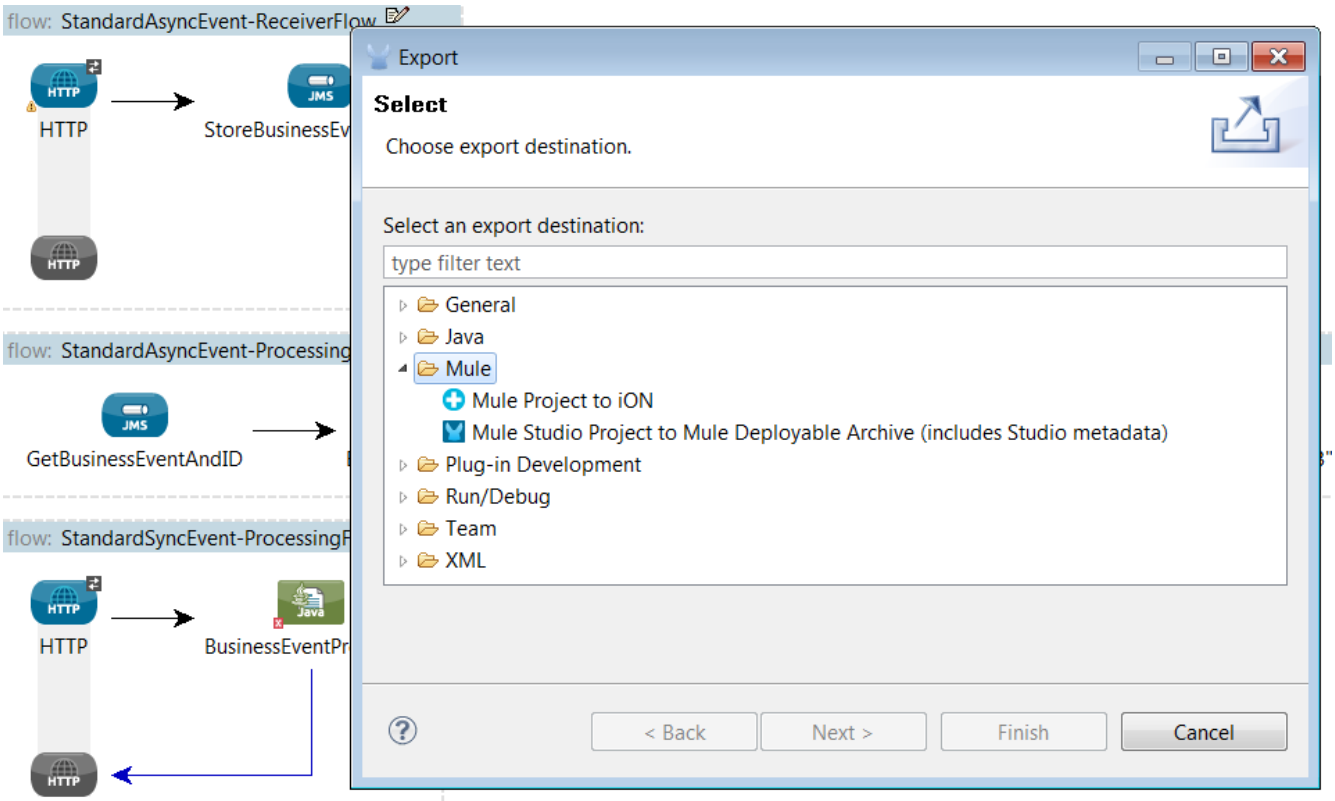
A developer or an architect, working in collaboration with a business analyst, can provide this knowledge by using the Data Attribute Update screen, provided below.

The Data Attribute Update screen below includes the section for an architect/developer who can provide the retrieval and validation methods for the selected data attribute.



If Mule is used, after testing the implementation of a specific business state, an architect or developer can place this implementation in the Mule Studio to support the Business Event Processor in the Standard Event Processing Flow.	

Then, with the Export feature of the Mule Studio, Mule Studio Project can be compressed to Mule Deployable Archive and imported in the Mule run-time cluster environment.



The bottom line: semantic approach makes all components of business and software development simpler and more efficient, provides immediate benefits to subject matter experts (SME), and helps engaging SME into collaborative work with IT while focusing on information and expanding initial semantic model. Workflow system development, the most common delivery mechanism for business processes, was just one of the examples. Another example is related to IT troubleshooting, when we try to use isolated system alerts to detect a critical situation before a customer would call with a problem.

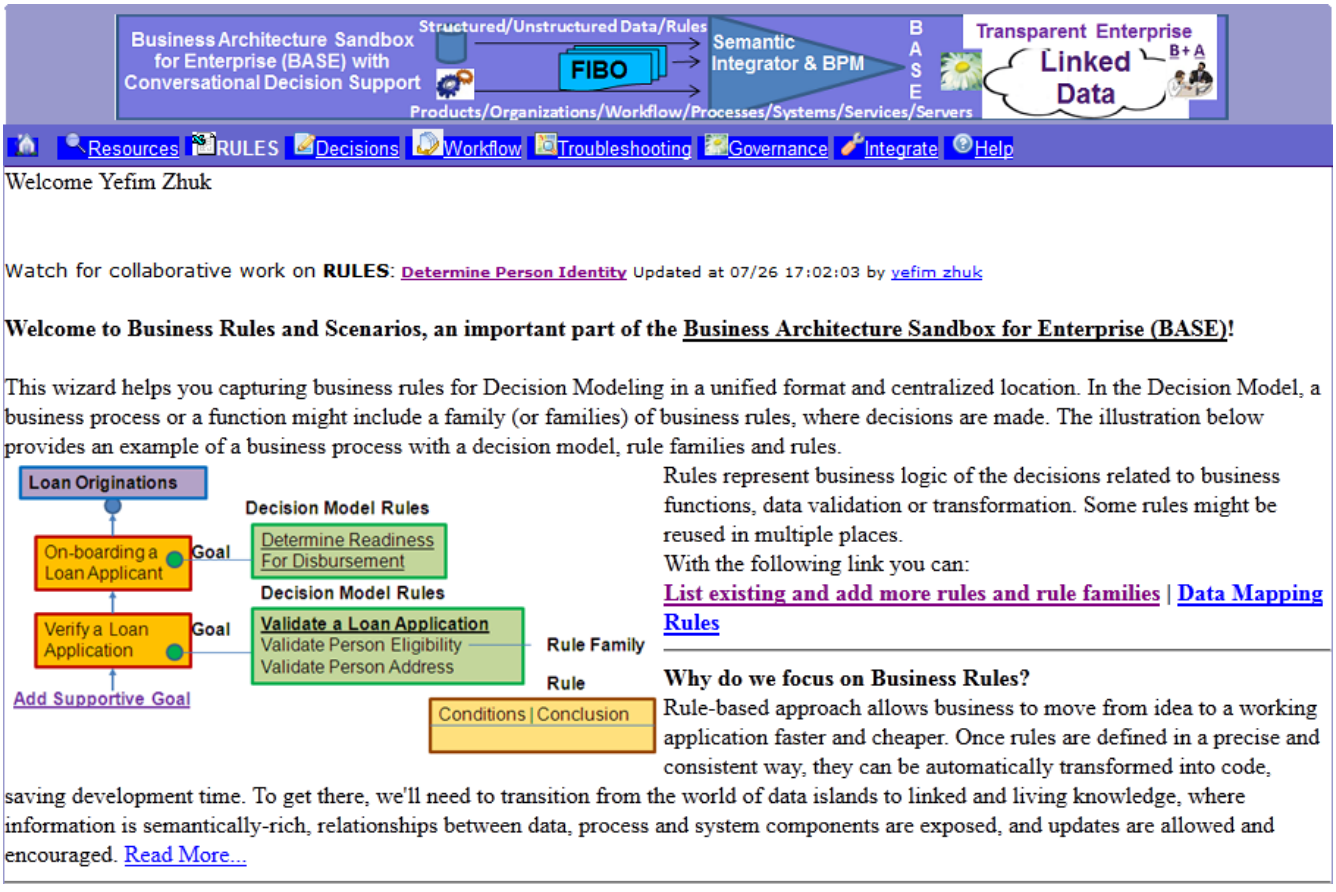
Collecting alert stories into a critical situational description

It is not unusual for any enterprise IT system to produce hundreds alerts each day or even each hour.

Each alert tells a small story and it is a very common practice to ignore these stories as they are not critical and the number of the alerts is overwhelmingly big.

Yes, thousands of alerts produced by expensive monitoring systems are usually ignored, until a customer called in and complained about a real problem.

The Semantic Model in the BASE allows a business analyst to connect multiple alert stories into a situational description and detect a coming problem.



The rules can be created and reused across multiple applications. One of the benefits of the semantic approach is its focus on information with the significant shift to providing the knowledge of handling data together with data versus traditional business logic placed in multiple applications dealing with data. This approach removes the biggest problem of using rules engines: the struggle of supplying data for the rules. The concept and its implementation were described before in the section of “data know how to handle data”.

The Business Architecture Sandbox for Enterprise (BASE) helps to collaboratively create, govern, and run rule-based decision models. Once rule families and rules are defined in a precise and consistent way, they can be automatically transformed into code, saving development time. The rules and rule families can be potentially reused by several components in different business processes. The name for a Rule Family usually reflects a conclusion portion of the rules. You can introduce new and edit existing rule records in the Rules Playground and Repository. For Service Integration the BASE can help creating Data Mapping Rules for interfaces.

List all rules | List all rule-based components | Selected Rule Family is Determine Critical Situation (id#5)

RuleFamilyId	RuleFamilyName
5	Determine Critical Situation

ConditionDataNames:
Application File Name Find Best Match or Create
Application Log Pattern Find Best Match or Create
Number of Events Find Best Match or Create
Time Duration Find Best Match or Create

ConclusionDataNames:
Alert Action Find Best Match or Create

Edit Record | Delete Record | Conditions and conclusions | History | Export

Add more records

In the example above, the rule family represents a description for a critical situation. Run-time analysis of the application log files takes into account specific alerts from several applications with their static and dynamic parameters, including text patterns and number of events during specific time slots.

Each rule is represented by a decision table below where Conditions and Conclusions provide a single row. Each Condition in the rule includes a Data Attribute, a conditional operator, and a value of data attribute. The same three components are present in a Condition. Usually it is a single conclusion in the rule with the Operator "Is" or "Equal". The Data Attribute, like Person Credit Score, might be known to Data Dictionary or can be created on-the-fly.

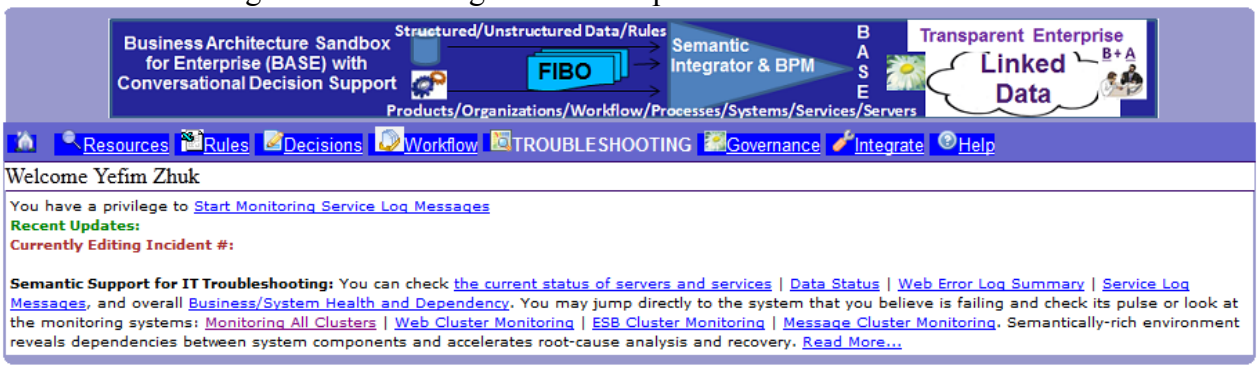
List all rules | List all rule-based components | Selected Rule Family is Determine Critical Situation (id#5)

Conditions				Conclusions	
Application File Name Edit/Delete Data Name	**Application Log Pattern** Edit/Delete Data Name	**Number of Events** Edit/Delete Data Name	**Time Duration** Edit/Delete Data Name	**Alert Action** Edit/Delete Data Name	
Equals Text Pattern VIIO Edit Condition	**In Text Patterns** Failure in calling Precise ID Edit Condition	**MoreOrEqual** 3 Edit Condition	**LessOrEqual** 3 min Edit Condition	Message: VIIO fails in calling Precise ID. This will impact the [Private Credit Originations] Process. Recommendations: contact VIIO-vendor and [OLA] to narrow down and fix the problem. Edit Conclusion	Delete Rule
Equals Text Pattern Account Origination Edit Condition	**In Text Patterns** Timeout Edit Condition	**More or Equal** 2 Edit Condition	**Less or Equal** 3 min Edit Condition	NotifyOnMessage(Message + " Origination Timeout") Edit Conclusion	Delete Rule
Add Condition	Add Condition	Add Condition	Add Condition	Add Conclusion	

Two rows in the decision table constitute two rules. The first rule will produce a message, when the VIIO application sends the alert with the text pattern “Failure in calling Precise ID” 3 times or more during 3 or less minutes.
The second rule is concerned with the Account Origination application, which sends the alert with the text pattern “Timeout” 2 times or more during 3 or less minutes.
The notification about a critical situation will be produced in the case when the first rule creates a

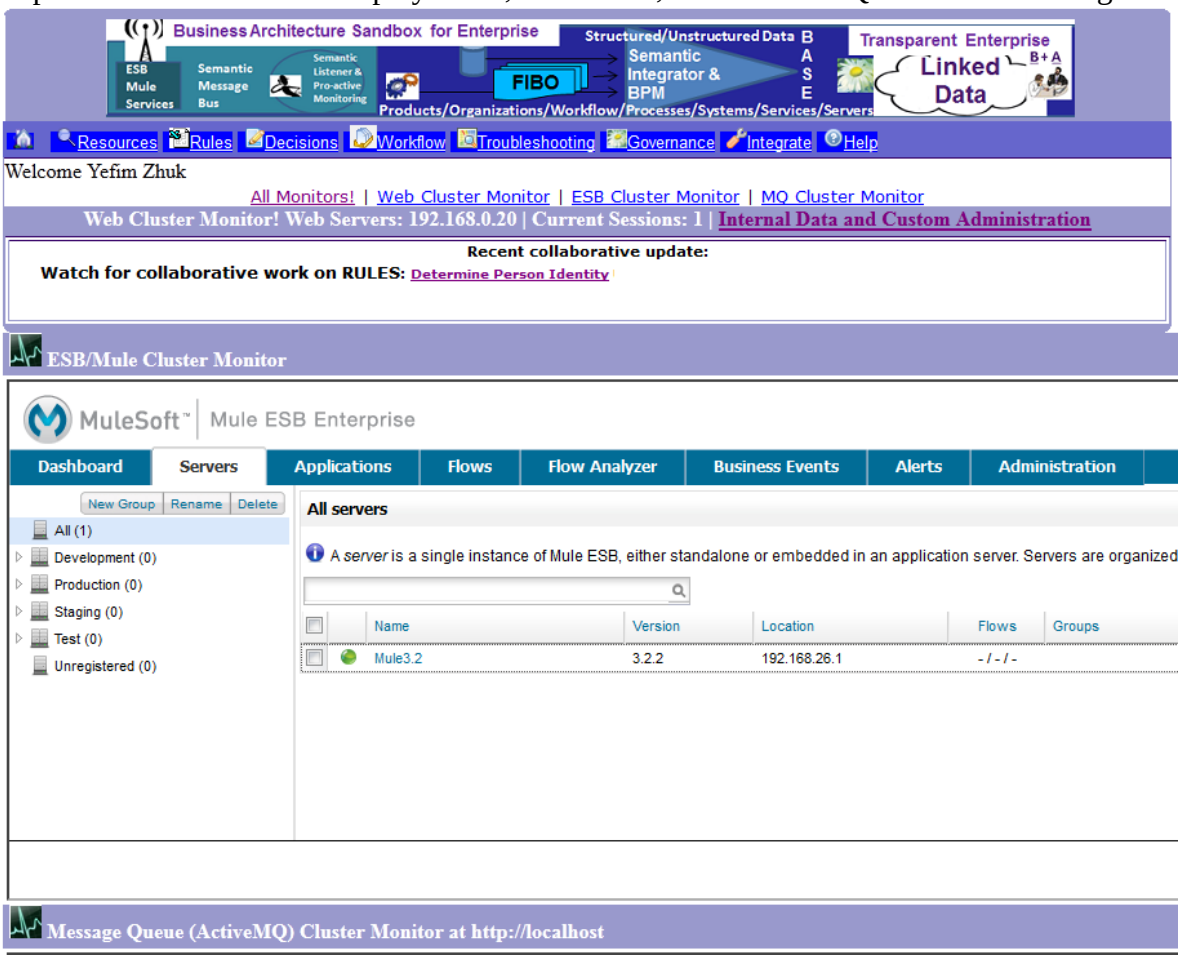
Message object and the second rule confirms all rule conditions.

The BASE provides several options in the Troubleshooting mode with the common goal to enable collaborative root-cause analysis with the ability to capture the results of the analysis with the extended semantic model and the rules in the decision tables. This is one more side of the multi-dimensional effort of transferring “tribal knowledge” into more precise forms of the semantic world.



The attractive side of this story is in its direct connection to existing technologies. There is no technology gap. Semantic seeds are reasonable placed in the SOA ground for further standardization.

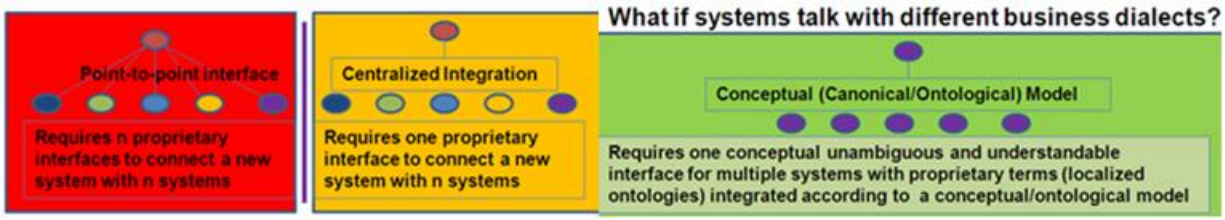
With the integrated monitoring facilities, semantic model allows developers to track dependencies between business and supporting system components, directly relate failing services and their business impact. The screen below displays Web, Mule/ESB, and Active MQ cluster monitoring.



Standard event processing and clustering solutions for high availability and failover, as well as troubleshooting automation will decrease dramatic maintenance efforts by IT and help shifting focus to information, primary task of IT.

Prepare for multiple partners and business dialects

The illustration below tells the story of the integration evolution, from point-to-point to centralized, and further to canonical interfaces with the semantic layer, which connects multiple business dialects.



This semantic layer will provide mapping of proprietary data to Canonical Data Model (Common Ontology) language. This is an important component of connecting the systems. This is also essential for designing API for 3-rd party developers.

	The semantic layer on the top of ESB will change the way of handling enterprise messages. This layer will allow developers to introduce a **semantic listener** and provide opportunities for subject matter experts to talk business terms while expressing their interest in specific reports based on enterprise messages.

This is another step in the right direction: preparing a semantically-rich enterprise environment.

Semantically rich enterprise environment

It is amazing how much can be accomplished with consistent and meaningful application messages.

The most common application messages include service calls, diagnostics, and error reporting (logging) information.

By providing meaningful service names, descriptions, and messages, developers create semantically rich application environment.

Service Names and Descriptions

Service name must reflect its purpose and should be readable in English. Service name usually consists of two or more concatenated words, like FinalPayment, etc.

Services are designed for reuse by several applications. **Service calls will become the language that multiple applications share** across the enterprise and with the partners. Commonly accepted vocabulary will consistently serve as a formal interface in the process of creating new artifacts, from document and subject names to the names of application services and their methods or operations.

Canonical Data Model (CDM) and Common Vocabulary (Ontology) help developers to name the services in a non-ambiguous way, so the service names can be understood by people as well as computer systems.
One example of a direct interaction between business, developers and ontology is provided below.

EXAMPLE:

The setFinalPayment() operation/method will be defined in the FinalPayment service. (The FinalPayment is one of the existing concepts in the Common Ontology.)

By sticking to the names existing in the Common Ontology, developers, architects and business analysts will come closer to a common language that is the key in improving business efficiency. There are multiple cases when similar service calls or operations differ by their parameters and return values.

In such cases the same operation names can be used and the technical differences can be reflected via the request and response objects passed as parameters with the service calls.

Service descriptions are necessary part of any service catalog or service registry.

Service descriptions include: the service layer name (Business, Utility, or Data layer), business specifics (if any), for example, "data layer/Collections" or "utility" (utility services are often called enterprise services) and 4-10 lines describing the service from a functional point of view. A composite service description will include a brief list of the services directly called by this service.

Service usage information will include a list of known consumer-applications that will call the service and (optional) service level agreements for each usage.

Service access information includes the endpoint and messages.

Service management information includes description of the business value providing by the service with the related service management specifics, which help to control and demonstrate this business value.

Service monitoring, diagnostics, and error reporting (logging) messages

In a semantically rich environment, there is no need for complex monitoring tools. The service names and descriptions as well as application messages are self-explanatory and directly tied to the execution model.

Application messages can be done in the style below, where elements of the message represent an optional subset from the set of elements below.

Each message can include as many properties as necessary with the property name before the value. The message should clearly provide the following information: WHEN (time), WHAT (description of

the failure), WHERE (system or/and service name), HOW Serious (type), HOW to fix (recovery action), WHO should be notified.

[[time]: currentTime], [[application]: BestVendorApplication], [[action]: com.its.actions.Customer.CustomerEnterOrder], [[type]: failure], [[reason]: database is down], [[recovery]: restart database], [[notify]: currentlyOnCallList]

Each part of the message is clearly framed by the [[name]: value] – pairs, which makes it very easy for processing with a semantic message listener.

A very simple "Semantic Listener" can understand and act upon these messages.

This approach, when it is consistently used across the company and industry, will create smaller, smarter, and less expensive semantic sensitive tools to monitor and manage service operations. The same message will become a valuable record in the root cause analysis and recovery processes. Such records can be RDF-formatted to and processed to compose the “situational awareness” factors.

Implementation can be done via log4j and a special “formatting” object which can be easily reconfigured. For example, at some point we can reconfigure the formatting object to generate RDF messages. A primary target for these messages will be a program, not a human being.

Semantic Technology is the only Promised Land to survive data storms

The greatest demand for Semantic Technology is in Content Management. The ocean of information is growing faster than our navigation facilities. Traditional approach of handling more subjects and relationships with more data tables proved to be wrong. Semantic Technology is the only Promised Land, where we can survive data storms.

Following the semantic approach, any story, event, report, or a published article is described as a simple graph where informational elements are connected by their relationships.

In real life everything is inter-connected, but in our descriptions of real life, we luck majority of these connections. One of the most important parts of content management is providing these connections and relationships with semantic content enrichment.

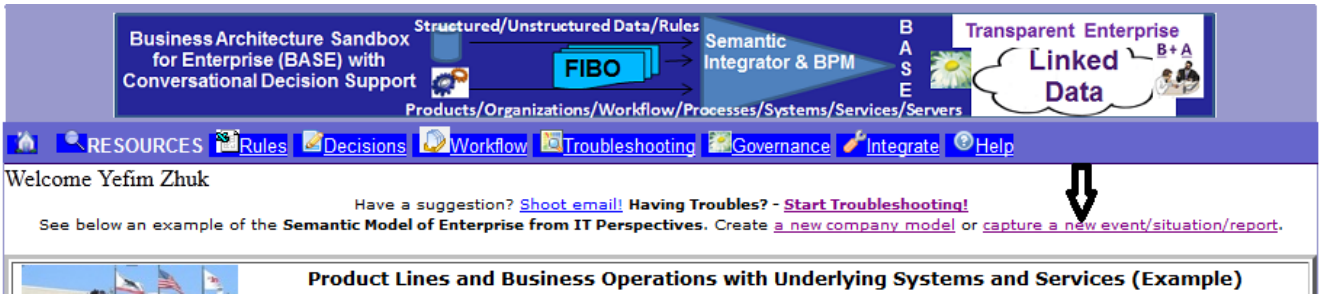
Quickly growing information graph requires special handling with Big Data tools, like Hadoop [7], HBase [8], and more.

One of the most ambitious ideas of the semantic approach is making a computer our partner in digesting information and making decisions. To accomplish this, we must provide to a computer program a very rich semantic background, enormous ontology, which would allow a program, like knowledgebase, add a new concept by connecting one to the existing background. That’s how we,

people, learn too. We are very limited in the common ontologies that can provide such a background. The biggest one is a “common reasoning” ontology from Cycorp, Inc. [9], another popular ontology is Dublin Core [10], and there are a few others. As to specialized industry ontologies, which can grow out of the common base, - this work just started. Probably, Financial Industry Business Ontology (FIBO) [11] will be the first standard ontology supported by the Object Management Group (OMG).
Several companies presented their work on these challenges at the International Conference on Semantic Technology in San Francisco, 2012 [12]. For example, “Migrating the LexisNexis® Content Management System”, and more.

When a specialized ontology is in place, you can use a set of powerful semantic tools, like Fluid Operations [13], or Allegro Graph [14], to open new horizons of managing information resources from company to industry levels.

The BASE helps content providers capture events, reports, or stories with auto-transformation into a semantic graph.



By selecting the option “Capture a new event/situation/report”, a “reporter” brings a conversational wizard screen displayed below.

The program will store captured information as a semantic graph, consistently building a linked data cloud.

ID	Type	Name	Description
01	Event	SemTech 2012	Semantic Technology Conference
02	Company	LexisNexis	Content Solutions and Services
03	Location	San Francisco	A dream city in North California, US
04	Time	June, 2012	June 3 – June 8, 2012
05	Story	Migrating Lexis	LexisNexis Content Management
06	Person	Stephen	Senior Architect at LexisNexis

ID	Relationships	Related ID
02	Participated In	01
01	Located At	03
01	Conducted During	04
05	Presented at	01
05	Presented By	06
06	Works at	02

While the BASE helps transition to semantic cloud architecture, a fast growing set of the powerful tools like Fluid Operations [13], Allegro Graph [14], Knoodle [15], and more, is making the way to standardize semantic operations in this Linked Data universe.

Summary

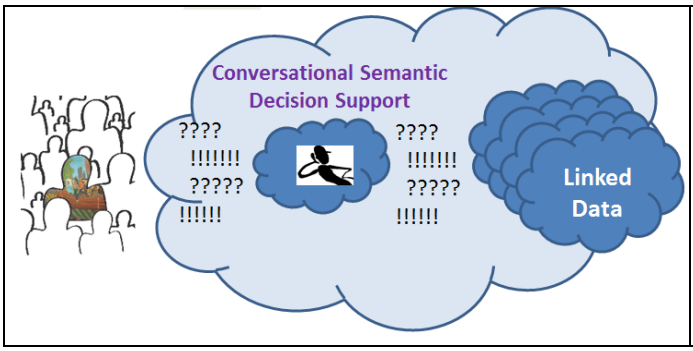
The gap between complex realities of the current enterprise infrastructure and Semantic Cloud Architecture seems so big that most of the companies are very cautious in approaching this cliff. The article offers very practical "baby steps" to transition to the IT of the future without upfront investment. The discussed approach is gradually placing the seeds of semantic technology in the current business ground, further standardizing business event processing, and establishing a self-sustaining process of IT transformation.

So, what are the major benefits of Semantic Cloud Architecture?

In Enterprise Data Integration: Semantically integrate scattered data in a unified platform for knowledge management [16], leading to streamlined business development [17] with less layers and better information focus.

In Content Management, Enrichment, and Analysis: Value-add by linking to free Linked Open Data sources; Simplified Publishing and Sharing of Data; Increase accessibility for new integration and partnerships; Open new horizons in collaboration with computer systems [18] on information analysis, discovering hidden dependences, and making mission critical decisions.

The bottom line is Enormous Cost Savings, shifting the focus of IT from Technology to Information, and offering Promised Land in the growing data storms.



Unified semantic information landscape simplifies the storage and management solutions to Linked Data and Conversational Semantic Decision Support (CSDS) systems.
CSDS help computers to better understand us by asking more questions [1] and use combined power of Semantic Technologies and Big Data to pacify the waves of the ocean of information.

Integrated software and knowledge engineering [2, 19] is transitioning from science fiction to science and IT of the future with Semantic Cloud Architecture opens new horizons and business opportunities.

References:

1. From Business As Usual to Knowledge-Driven Architecture, Jeff (Yefim) Zhuk, http://semanticweb.com/from-business-as-usual-to-knowledge-driven-architecture-part-i_b21243

2. Integration-Ready Architecture and Design, Jeff (Yefim) Zhuk, Cambridge University Press, A book on Software and Knowledge Engineering

3. Liferay, Open Source Portal, http://liferay.com

4. Mule-ESB, http://www.mulesoft.com/mule-esb-open-source-esb

5. Apache ActiveMQ, http://activemq.apache.org/

6. W3C on Large Triple Stores and http://www.w3.org/wiki/LargeTripleStores

7. Hadoop, a Big Data Apache project, http://hadoop.apache.org/

8. HBase, a Hadoop database to handle Big Data, http://hadoop.apache.org/

9. Cycorp, Inc., a common reasoning language, ontology, and inference engine, http://cyc.com

10. Dublin Core, Meta Data Initiative, offers common ontology, http://dublincore.org

11. Financial Industry Business Ontology (FIBO), initiated by EDM Council, www.edmcouncil.org

12. Semantic Technology 2012 Conference, San Francisco, http://semtechbizsf2012.semanticweb.com

13. Fluid Operations, Inc., Open Platform for Linked Data Solutions, http://www.fluidops.com/

14. Allegro Graph, RDF DB and a set of semantic tools, http://www.franz.com/agraph/allegrograph/

15. Knoodle, Content authoring and management solutions, http://www.knoodle.com

16. Rules Collector, Yefim Zhuk/Boeing, US Patent, The system to transform “tribal knowledge” to rules and rule-based applications

17. Knowledge-Driven Architecture, Yefim Zhuk, US Patent, The methods, architecture, and system to streamline business development and drive business processes with business rules & scenarios

18. Adaptive Robot System with Knowledge-Driven Architecture, Yefim Zhuk, US Patent, Robot’s teamwork with on-the-fly translations of situational requirements into adaptive robot skills

19. Related publications and demo pages