

Institute of Architecture of Application Systems  
University of Stuttgart  
Universitätsstraße 38  
D- 70569 Stuttgart

Bachelor Project  
(Studienarbeit Nr. 2185)

# Web-based Editor for WS-BPEL Processes

Bruno Miguel Castanheira Colaço

**Course of Study:** Computer Science

**Examiner:** Prof. Dr. F. Leymann

**Supervisor:** Dipl.-Inf. O. Kopp

**Commenced:** April, 15<sup>th</sup> 2007

**Completed:** December, 17th 2007

**CR-Classification:** C.2.4, D 2.3, H.4.1



# Abstract

Organization of a business project is something essential nowadays.

In this context, especially for complex applications, the use of a workflow has a key importance and is becoming more and more common.

This workflow is the architecture overview, which describes the interaction between the parties in stake in a business model.

The appearance of models BPM (1) that offers a graphical representation for a better understanding is something unavoidable.

On one side, BPM (1) model is more user friendly, but on the other side, the generation of a XML base language offers interoperability and easy computational processing.

The use of tools for modelling the BPEL is nothing new.

The goal of this project is to create a new plug-in for the web Oryx Editor (a BPM editor designed by the Hasso-Plattner Institute in Potsdam, Germany) in order to generate a WS-BPEL document, as well as to understand his limitations.

Another goal of this project is to create a plug-in that supports or can be able to be extended to BPEL4Chor.

# Table of contents

Index of Figures .....	1
1 Introduction .....	2
2 Editor .....	3
2.1 Stencil Set .....	4
2.2 SVGs and the Editor .....	5
2.3 Description of JSON elements for the Stencil Set .....	7
2.3.1 Callback Functions .....	8
2.3.2 Properties .....	8
2.3.3 Rules .....	9
2.4 Data management .....	11
2.5 Shape Design (limitations of the editor) .....	11
2.5.1 Bugs Found .....	12
3 Overview of BPEL .....	16
3.1 Context .....	16
3.2 Brief History of BPEL .....	17
3.3 The BPEL language .....	17
3.4 Abstract and Executable WS-BPEL .....	19
3.5 Relationship of BPEL to BPMN .....	20
4 Evaluation of the Editor with respect to BPEL .....	21
5 Implementation .....	22
5.1 Architecture overview .....	22
5.2 Representing the WS-BPEL .....	23
5.3 List of the shapes .....	23
5.4 Restriction with the use of roles .....	27
5.5 Overcoming some limitations of the Editor .....	29
5.5.1 Order of Execution .....	29
5.5.2 Presenting the WS-BPEL XML .....	29
5.5.3 Creation of new Fields .....	30
5.5.4 Properties and Shapes .....	33
6 Examples .....	41
6.1 Loan Approval .....	41
6.2 Travel .....	43
7 Conclusions .....	45
8 Future work .....	46
8.1 Creating the missing elements .....	46
8.2 Loading definition files .....	46
8.3 BPEL4Chor .....	46
8.4 Validation of the process .....	47
8.5 Loading an external BPEL .....	47
9 Appendix .....	48
Appendix A.1 .....	48
Appendix A.2 .....	55
Appendix A.3 .....	59
Appendix B.1 .....	61
Appendix B.2 .....	63

Appendix C.1 .....	66
10 Bibliography .....	74

## Index of Figures

Figure 2-I: Oryx Tool Bar .....	3
Figure 2-III : Before: Operation Allowed.....	14
Figure 2-IV: After: Operation Denied .....	14
Figure 2-V: Incoming Edge Bug .....	14
Figure 3-I: BPEL and WSL interaction.....	18
Figure 5-I: Editor of Complex Type.....	30
Figure 5-II: Search Field.....	31
Figure 5-III: Properties frame of Invoke .....	34
Figure 5-IV: catch handler in Invoke Activity .....	34
Figure 5-V: if example .....	37
Figure 5-VI: IfHolder properties .....	38
Figure 5-VII: if condition properties .....	38
Figure 5-VIII: Flow Example .....	39
Figure 5-IX: sequenceFlow properties .....	40
Figure 5-X: Flow properties .....	40
Figure 6-I: Loan Demo .....	41
Figure 6-II: Travel Flow Process.....	43
Figure 6-III: Travel process.....	44

# 1 Introduction

A workflow in the organization of a business project nowadays is something essential and is becoming more and more common.

Business Process Management models (BPM) (1) offer a graphical representation and, as such, are something unavoidable for a better understanding. BPM (2) model is more user comprehensive than the generation of a XML based language, which offers interoperability and easy computational processing. The use of tools for modelling the BPEL is nothing new, even in the frame of this project whose goal is to create new tools, for the Oryx Editor (a BPM editor) in order to generate a WS-BPEL (2).

In this project, it is our aim for the creation of the stencil set and the BPEL plug-in to do the less number of changes as possible in the editor core, so that we can analyse, as better as possible, the editor itself and the implications in the creation of the stencilsets, plug-ins and his properties. In this paper, we will give an overview of the editor *Oryx*, describe the architecture of our plug-in, our choices in matter of designing an extension of it in order to generate a WS-BPEL process file.

## 2 Editor

The project "B7 - Browser-based Business Process Editor" (3) of the department "Business Process Technology" of the Hasso-Plattner-Institute in Potsdam, Germany was about developing a web-based application for modelling processes with graphical notation languages. The referred project focuses in a front end application, a web client application.

The Oryx editor is a web based application that uses javascript, prototype (5), ExtJs (6) and SVG (7). It is based on three different layers where the data are held.

The first layer is the top layer, common to the core and the plug-ins, where our plug-in, the BPEL generator, will perform the designated operations with the values of this layer. The data are available under the form of Javascript objects, this will share their lifetime with the shapes visible on the canvas.

The second layer is the data storage, the DOM: this layer will hold the data transmitted by the server. The last layer is the persistence of the data on the server.

The approach given to this editor was his extendibility and flexibility, the appearance of plug-in tools, to extend the editor's functionality and a definition of a custom "stencil set" was, and is, one of his highest achieves.



*Figure 2-I: Oryx Tool Bar*

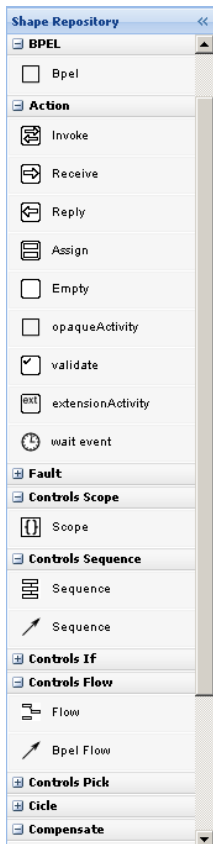
It was necessary to add a plug-in embedded into the Oryx editor, whose icon we can see in the Figure 2-I, red circled.

The stencil set specification was designed in order to be able to implement all constructs of the Business Process Modelling Notation (BPMN) (8) without using BPMN dependent approaches in the specifications that are only useful for this stencil.



## 2.1 Stencil Set

On the right side of the Oryx editor, a multiple choice frame is there to show a list of the



shapes available, ordered by groups, according to their function.

All these shapes are created thanks to their definition in the “stencil set”. The main idea behind it is a definition of graphic objects and rules that specify the relations among the different graphical object.

Such objects can be labelled as “node” or “edge”. Nodes are typical central elements, edges are normally links that connects two graphical objects. Each of the above referred graphical objects can contain properties, that can possess some dependencies to other shapes or that can be used later by Oryx extensions, in our case a BPEL generator. Those properties can also manipulate the graphical representation, by the dependence made with the SVG elements.

Figure 2-II: Shape repository

The technology used for the description of a stencil set is basically a JSON (JavaScript Object Notation) (3) file with the descriptions of the elements (rules, properties and the SVG file associated). The roles hierarchy is also described in this file, as well as some call back functions.

The loading of a stencil is made in the element “<a/>” with the property *href* to the referred “*json*” file and the value “*oryx-stencilset*” for the property “*rel*” that contains the “*link type*”.

The stencil set can also be loaded dynamically, with the use of one plug-in (“*Add Stencil Set*”), since we are restricted by the browsers for security reasons, being only possible to load it from Internet.

The use of a namespace is mandatory. It is necessary to define the top element, which is the object that belongs to the canvas. This can be done with the definition of a “meta” element (a HTML element) with the property ‘*name=“oryx.type”*’ and with its content referring to the namespace plus the character “#” and the *id* of the element defined in the *json* file.

## 2.2 SVGs and the Editor

As referred above, there is a direct relationship between the *json* file and the graphical representation. Not only each element needs to refer to a certain SVG, but the properties can refer also to a certain sub-element in the SVG file, being possible to change his value dynamically in the editor.

The document SVG is a XML document that represents graphical vectors. Since this is a XML element, can be used and even manipulated directly in the Oryx editor. To be able to declare specifications, like resizing, and the definition of certain elements that belong to the Oryx, a custom namespace is defined.

Not all of the specifications are supported in this version of Oryx, since there are some restrictions with some browsers. Internet Explorer (3) requires the installation of a plug-in “Adobe SVG Viewer” (3), to support SVGs (4), Safari 2 (5) and Opera (6) does not support SVGs, the Browser Firefox Mozilla 2.0+ (7) are left. Being Firefox the main target as platform, the development of the BPEL generator was made under this one.

It is just the direct relation between the properties and the SVG element that makes the design dynamic. In the editor it is possible to change text and colour, as well as hiding or showing some nodes, all of these actions (in the SVGs elements) are made according to the value of the property. In the SVG elements some properties, or elements, can be defined so that they do not belong to the SVGs standards.

Because of that, it is necessary to specify namespace for further use.

In this paper, we will describe some elements that were used in this project, and some the most relevant importance. For additional information please check the Oryx documentation. The property “*oryx : resize*” is part of a shape element, like ‘*rect*’ or ‘*path*’, and can hold the values “*horizontal*” or “*vertical*” defining whether the resize is possible. The definition of a minimum and maximum size is defined in the ‘*g*’ element with the properties “*oryx : minimumSize*” and “*oryx : maximumSize*”, the non definition of this means that there is no minimum and maximum values.

The Oryx properties for the text elements (in the SVG document) are the ‘*rotation*’, that enables the rotation of the text, and the ‘*align*’ to align the text in horizontal and in vertical (with the values “left, center or right” and “top, middle or bottom”, respectively).

Another concept introduced in the SVG element, is a definition of special node, called *magnet*, to where you can dock other nodes or edges and connect them.

You can connect a *docker* to any point on a node, but magnets help the user, creating nicer looking models. If you do not define a magnet for a node, it will have a default magnet in the centre. Another element introduced is the docker, that is a control object that can connect an edge to a node or, like in this case, to connect two nodes. A node can have no more than one docker that can be defined by using a ‘*docker*’ element from the oryx namespace.

The SVG that describes the edge is very similar to the one described above. With the difference that, this one uses the *defs* element in the SVG document. With this definition it is possible to put the paths custom drawings in their start, end and middle.

## 2.3 Description of JSON elements for the Stencil Set

As previously stated, the stencil set is a definition in a JSON (4) file of a set of elements and properties that composes the stencil. Here we are around to give a brief description of the stencil set and to describe some relevant properties. For more definitions, please consult the stencil set description of the Oryx specification (11).

Since the editor uses a grid object from the ExtJs API (5), the objects for properties are also available (in this chapter we are not going to describe the needed requirements to create new property, this information will be discussed later). For the type of the properties, the objects available are those from the ExtJs Library: objects that support String, number value, Date, or Color are some of them.

Each file has a custom title, namespace, description, a set of stencils and a set of rules. In the “*stencilset*” each element requires a set of based properties like a unique “*id*”, a “*type*” with the value of ‘*node*’ or ‘*edge*’, the “*svg*” of the current element ( this SVG will be the graphical representation), and icon for the stencil set tool.

The set of Roles is also given to the element, the role that has the same name of the element, if exists, is also given. The properties “*Serialize*”, “*Deserialize*” and “*layout*” will hold callback functions. The “*Serialize*” function of a stencil is called, whenever a shape is persisted into the DOM. The values of all properties are serialized automatically, but if more information is needed to be serialized, it is possible to add information in the array ‘*data*’ parameter of the function. This object added in the data array needs to have the properties ‘*name*’, ‘*prefix*’, ‘*value*’, and ‘*type*’.

The ‘*prefix*’ is the eRDF (6) schema and ‘*name*’ is the identifier for that data in the schema. The property ‘*value*’ contains the value you want to add. The ‘*type*’ property can either be ‘*literal*’ or ‘*resource*’. It is needed for the internal eRDF data format. In short, ‘*literal*’ sets to store the value as a literal, whereas ‘*resource*’ defines the value to be a reference.

### 2.3.1 CallBack Functions

The *'deserialize'* is called before Oryx processes the serialized data, so it is possible to manipulate before the editor analyse it. This function is normally the reverse of the *'serialize'* callback.

The *'layout'* function is the responsible for the custom bounds of the shape, this one is called every time the shape is moved or dragged. The use of this callback function will call the same layout function of the above shape hierarchy.

### 2.3.2 Properties

*"Properties"* set is an attribute from the shape element (in the JSON file) composed by an array with several properties. Each property, depending on his type (also an attribute), requires some specific attributes. Here we will give an overview of these requirements, for more specification please refers to the Oryx Specification (7).

Each property requires also a unique id, in the shape element, a title, an initial value, and a description, his mode being read only or not. The interaction between the SVG shape and the property can be made using the property *"refToView"*, this value will represent the id in the SVG document. The actions made, changing of colour, text value, enable, etc., will relate the type of the property to the SVG representation.

In *"type"*, as mentioned above, there is a dependency between the Extjs Library and the types offered by the Oryx editor as properties.

### 2.3.3 Rules

In the editor several kinds of rules were introduced, defined in the stencil set as well as a set of roles for each shape.

Rules are *cardinalityRules*, *containmentRules*, and *connectionRules*.

There are also two callback functions: *canConnect* and *canContain*, that are available for specific validations where it is not possible to be described by the available rules. Because these functions are shared among of the *stencilset*, it is necessary to check the *id* of the calling shape. These functions will only be called in the case of the action being available, so that, with the analysis inside this callback functions, the action can be restricted or allowed.

With this callback functions it is possible to overcome some of the limitations of the editor, but nevertheless we propose/changed some of the behaviours of the rules descriptors in order to put it more dynamic, this subject is discussed further in the section 2.5.

Each shape contains a field to which is possible to assign several types of roles. Each shape contains in his set of roles one whose name is the *id*. Roles can be abstracted: it is not necessary it corresponds to a shape to be created and evaluated.

The '*connectionRule*' describes which role can have an incoming shapes and an outgoing shapes, the non-definition of some of these properties will describe the non-restriction of the same, being possible to connect the fields ( or the shape itself) that has no definitions, without any limitations.

In the property '*cardinalityRules*' is possible to define the number or occurrences with the field *maximumOccurrence* for each shape, that describes the amount children's that can occur in a parent. After the changes made in the Oryx Core, the restriction is now made under the role and not related with the child *id*. Another restriction that can be done in this field is to set a defined number of connections (in

incoming and outgoing shapes), this restriction is at the moment made by the id of the incoming/outgoing shapes.

The '*containmentRules*' describes a parent-child-relationship, the field '*contains*' will describe the multiple roles, where a shape can have one role in order to be appended.

As described earlier, an Extension of the rules can be done.

One of these extensions is with the use of the '*canConnect*' callback function. For instance, in our stencil it was used the connection of a sequenceFlow edge (that represents the <link> element), that can only be made if the activities to be connected are in a Flow shape. At this purpose, it is necessary to analyse the connected shapes to understand if there is a parent-child-relation, even not directly as a child, otherwise it is not possible to connect the same. This callback function is called every time there is a valid connection made, being possible to cancel the same action.

Likewise the callback function described before, the '*canContain*' is analysed each time a shape is appended into a new child, but only if they are allowed by the *candinalityRules* and *containmentRules*.

## 2.4 Data management

Oryx internal data format is based on the Resource Description Framework (RDF) or more precisely eRDF for embedding RDF into. Oryx only knows resources and literals that are accessed using triples with a subject, a predicate and an object. The subject is the resource you are interested in. The object is a literal or another resource and the predicate specifies the relationship between the subject and the object. With these triples you can easily build up a data hierarchy. The API to access those resources can also be used by other applications, for example a process execution engine (15).

## 2.5 Shape Design (limitations of the editor)

Being the Oryx editor and our *stencilset* designed for the Firefox (8) browser, it is not possible to access to other files besides the ones available on the Internet.

This restriction is due to a security policy, in the accessing of the *filesystem* objects. Because of that, the use of the browser to create a file is restricted. Being aware of this fact, we tried to overcome this issue using several techniques described in the chapter 5.5.2.

The documentation of the Oryx editor does not make a significant reference to the Ext Javascript Library (11), but the dependence with the referred API is relevant, since most of the layouts use objects from the referred API. The Grid and the properties are examples of it. Since not all of the properties available in the *Extjs* API can fulfil all the needs, if for instance is required a complex or just a different property, you must be aware that this new property needs anyway to fulfil some requirements in order to be considered a “*PropertyField*”. These fields will be described in the chapter 5.5.3.1 and 5.5.3.2.

There are some restrictions in the SVG files. For instance, it is not possible to define a ‘*def*’ element (<def/>) in Nodes shapes, but it is possible to make referred



definition in Edge shapes. This inconsistency is related with model approaches, even though in most of the cases it is possible to achieve the same goals with another approach (creation of other elements).

### 2.5.1 Bugs Found

In the version of Oryx which this work is based on (v.0.10), some bugs have been found. Some of these are not relevant, being more related with the model used, but others generate several restrictive problems, that need to be reviewed and fixed.

#### Character ‘<’ in properties

One of these bugs is the use of the character ‘<’. When it is used, the model saved cannot be loaded again. This fact is due to the storage of the referred character in the DB. So, when the process loads, writing the content of the DB in the HXML page, corrupts the XML and shows the *“XML Parsing Error: not well-formed”* of the current page.

#### Canvas Sliding

Other errors that we encounter in this version of Oryx are the shifting or sliding of the canvas. This behaviour happens in the drag and drop movement and is due to several long properties in each shape. Each shape, actually, represents a *“div”* in the DOM element, and this *“div”* is composed by several *“span”* elements, each one representing one property, as we can see in the Appendix C.1. So, when the content in each shape overcomes the length of the Oryx canvas, it occurs the slide in the drag and drop movement. The value of each property is under the span element and this fact does not allow the definition of the width. The trick was to minimize the dimension of just one property, instead of the entire shape, using a `<br/>` element to separate each span element. But this topic should be investigated more deeply, since the solution founded should not be used as permanent.

### **Optional field in stencilset**

There was a small bug related with this field. Being this one never considered, it was always with “*true*” value. In case of the non-definition of this property, this one should be meant as optional. Because of that, it was always changing his value to “*true*” even when it was defined as “*false*”. This bug was corrected, being now fully operational. Since we want to establish a relationship between this *optional* field and the requirement of filling the property, it was essential that this field was fully operational/supported .

### **Id instead of Roles**

One of our goals was to model the WS-BPEL (3) specifications as much as possible with the use of roles, in the stencilset. There were bugs related with the *cardinalityRoles*.

The *maximumOcurrance* describes the maximum number of the shapes with the referred role that can be contained in a parent. In fact, the dropped shape did not consider the other shapes roles, but their “*id*”. The best way to demonstrate this bug is with the following example:

For instance: if we have a shape ‘A’ and a Shape ‘B’ with the same role, Role ‘Y’.

The role Y has the *maximumOccurrence* with the value of 1, the supposed behaviour is that we can only have one shape A or one Shape B in the same Parent since is the role itself that has been restricted and not the shape, but, with the previous settings, we can have one of each, since is not the role that has been restricted, but the stencil shape name.

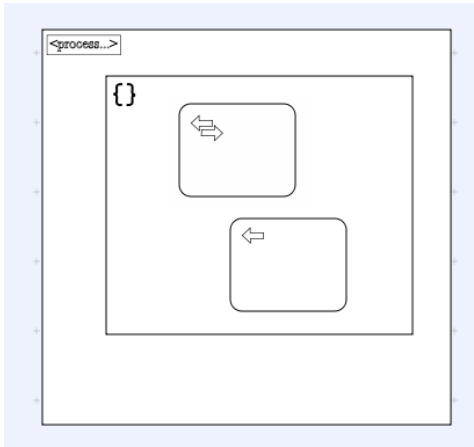


Figure 2-III : Before: Operation Allowed

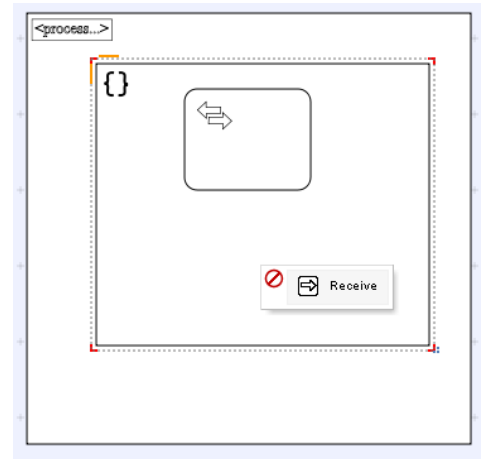


Figure 2-IV: After: Operation Denied

This bug has been fixed.

If a shape can have more than one role available, its roles will be checked one by one to see if there is still some with “space” available, or at least one whose property is *undefined* (in this case it can hold infinite shapes). A similar bug with the attribute *maximum* in the *outgoing/incoming* shapes of the cardinality roles, was found, unfortunately at this point it was not possible to correct this bug. The bug found has the same behaviour as the one reported before.

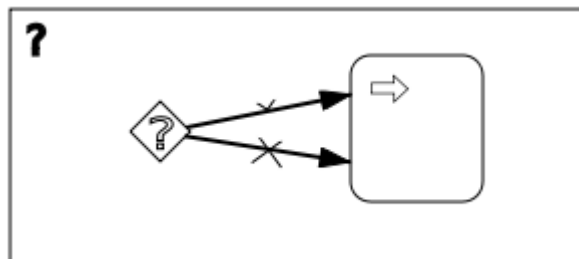


Figure 2-V: Incoming Edge Bug

In the above figure it is shown a receive shape that has been connected both to a “false” and a “true” if activity. This operation should not be allowed because there is a cardinality restriction to a role that is shared with both of the links.

```

{
  "role": "Activity",
  "maximumOccurrence": 1,
  "outgoingEdges": [
    {
      "role": "sequenceOrder",
      "maximum": 1
    },
    {
      "role": "sequenceFlow",
      "maximum": undefined
    }
  ],
  "incomingEdges": [
    {
      "role": "sequenceOrder",
      "maximum": 1
    },
    {
      "role": "sequenceFlow",
      "maximum": undefined
    },
    {
      "role": "sequenceIf",
      "maximum": 1
    }
  ]
}

```

In fact, as we can see from the code, the shape that contains the role *sequenceIf* can only be one. The check of “space” available is made under the “*id*”(of the shapes in stake) and not their roles.

### **Low case “*id*”**

In the documentation of the Oryx editor was not reported any restriction related to the need of non capital letter in the “*id*” of each shape. The use of capital letter brings to the loss of data. The reason is that the change to non capital letters is uniquely made when the data is received from the server.

Since is our interest to have the maximum relation between a stencil set and the BPEL (in elements, properties, etc), we changed the eRDFparser to avoid the change to non capital letters. However, since the reason for the use of the lower-case in the RDF is not known, this issue will be reported to the Oryx team, even though in this version of Oryx it is full working.

### 3 Overview of BPEL

The Business Process Execution Language for Web Services (WS-BPEL or BPEL for short) is an XML-based language for defining business processes that provides an interoperable, portable language for both abstract and executable processes was designed from the beginning to operate in the heterogeneity and dynamism that is commonplace in information technology today. This language specifies Web services-based business processes. It is an XML-based language which is built on top of WSDL (19), XPath (15), and XML Schema (21). Processes written in BPEL can orchestrate interactions between Web services using XML documents in a standardized manner. These processes can be executed on any platform or product that complies with the WS-BPEL specification (2). The WS-BPEL provides those missing process definition capabilities and it is considered as one of the key building blocks of Service-oriented architecture (SOA).

#### 3.1 Context

With the evolution of SOA and the adoption of Web services the role of services orchestration became more important. The ability to compose processes out of services provides flexibility in creating and building new applications. The prerequisite for this is, however, alignment with basic principles of SOA. Encapsulation of the business logic into independent, autonomous services which offer interfaces as the only means to interact with the service is seen as the basic principle of SOA. This implies that it is necessary to separate the business logic from the presentation logic and the process flow logic. While business logic would be encapsulated within Web services, the process flow logic would be part of a new layer – services orchestration.

### **3.2 Brief History of BPEL**

BPEL is a convergence of language features from IBM's Web Service Flow Language (WSFL) that uses a directed graph approach and Microsoft's XLANG, which is the orchestration language, used in BizTalk server and has a block-structured approach. Both WSFL and XLANG are now superseded by the BPEL specification (14).

BPEL 1.0 was jointly developed by IBM, BEA, SAP, Siebel, and Microsoft in August 2002. In April 2003, BPEL 1.1 was submitted to OASIS to obtain even broader industry acceptance and open standardization, and being the last version released in April of 2007, version with the name WS-BPEL 2.0.

### **3.3 The BPEL language**

A BPEL process consists of a top-level process element that can contain: variables, event handlers, fault handlers, compensation handlers, partner links, and a single (complex structured) activity. Variables are typed containers that hold data. They may be typed by using WSDL messages, XML

To define business processes, BPEL describes a variety of XML elements, such as:

- **Variables:** defines the data variables used by the process, providing their definitions in terms of WSDL message types, XML Schema simple types, or XML Schema elements. Variables allow processes to maintain state data and process history based on messages exchanged.
- **PartnerLinks:** defines the different parties that interact with the business process in the course of processing the order. Each partner link is characterized by a partner link type and a role name. This information identifies the functionality that must be provided by the business process and by the partner service for the relationship to

succeed, that is, the portTypes that the purchase order process and the partner need to implement.

- **faultHandlers:** that contains fault handlers defining the activities that must be performed in response to faults resulting from the invocation of the assessment and approval services. In WS-BPEL, all faults, whether internal or resulting from a service invocation, are identified by a qualified name. In particular, each WSDL fault is identified in WS-BPEL by a qualified name formed by the target namespace of the WSDL document in which the relevant portType and fault are defined, and the `ncname` of the fault. It is important to note, however, that because WSDL 1.1 does not require that fault names be unique within the namespace where the operation is defined, all faults sharing a common name and defined in the same namespace are indistinguishable. In spite of this serious WSDL limitation, WS-BPEL provides a uniform naming model for faults, in the expectation that future versions of WSDL will provide a better fault-naming model. (22)

The Figure 3-I:BPEL and WSDL interaction shows a relation mapping between the BPEL process definition and the WSDL.

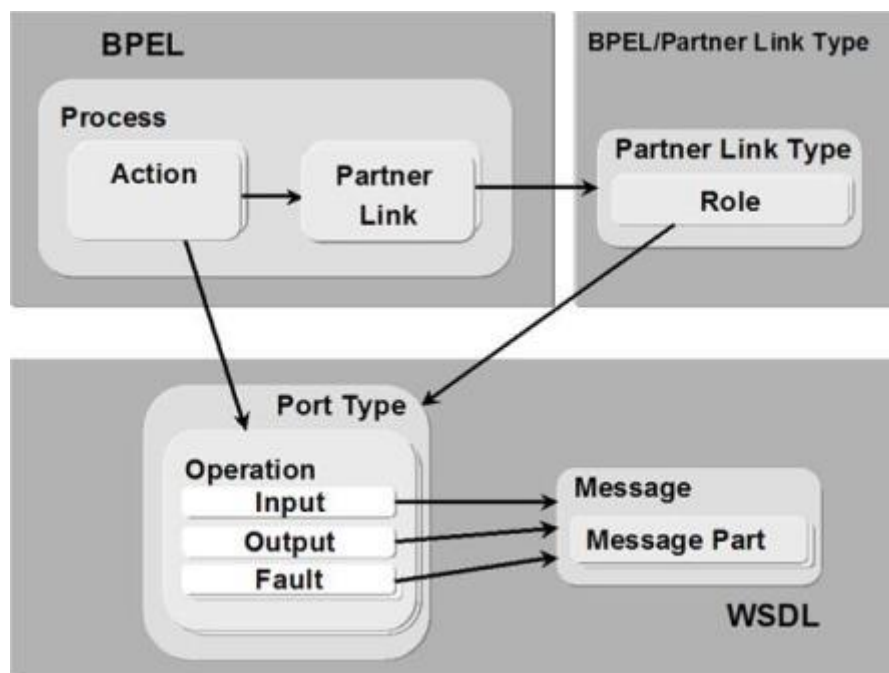


Figure 3-I:BPEL and WSDL interaction (14)

The life cycle of a process starts with the creation of an instance when the system receives a message that can be consumed by a receive activity and whose *createInstance* attribute has the value “yes”.

The process starts by activating its top-level activity. The process terminates when the top-level activity is completed, when the process throws a fault for which a handler is not found, or when a terminate activity runs. A BPEL business process interoperates with the Web services of its partners, whether or not these Web services are implemented based on BPEL. Also supports the specification of business protocols between partners and views on complex internal business processes (15).

### **3.4 Abstract and Executable WS-BPEL**

An *abstract* process provides a description of a related range of behaviours; one can think of it as representing a set of executable processes. Abstract processes have access to the same range of syntax and semantics as executable BPEL processes. Opaque tokens enable explicit hiding of information, and in some cases, may be omitted.

An *executable process* contains all the actual message data, operations, and partner information required for a running process. It uses the full power of data assignment and selection (15).



### **3.5 Relationship of BPEL to BPMN**

As an illustration of the feasibility of this approach, the BPMN specification includes an informal and partial mapping from BPMN to BPEL 1.1. A more detailed mapping of BPMN to BPEL has been implemented in a number of tools, including an open-source tool known as BPMN2BPEL (23). The fundamental differences between BPMN and BPEL, make it very difficult, and in some cases impossible, to generate human-readable BPEL code from BPMN models. Even more difficult is the problem of BPMN-to-BPEL round-trip engineering: generating BPEL code from BPMN diagrams and maintaining the original BPMN model and the generated BPEL code synchronized, in the sense that any modification to one is propagated to the other (15).

## 4 Evaluation of the Editor with respect to BPEL

As it was previously stated, there are some differences between the specification of the BPEL and the model used in the Oryx editor. In this section, we will describe some of the errors encountered, as well as some limitations of the Oryx for the BPEL.

Analysing the existing tools and the BPEL specification, we realise that there is a need of establishing an order of the elements. However, the Oryx model does not assure any dropping order, or even to change that order.

Since the Oryx is made with the use of the Extended javascript (EXTjs), the properties supported were the ones available by the same API.

Furthermore, creating different properties is something unavoidable, especially when you need to store a “complex” value in a property, or when is our interest to have some interaction between shapes.

In the BPEL specification is not allowed the use of whitespaces characters in names, namespaces, etc. However, the property field of type “*Text*” allows it, this being a restriction of the Oryx editor. We should be aware of this factor when we are modelling the process, or, the *whitespaces* characters will appear on the BPEL file.

## 5 Implementation

### 5.1 *Architecture overview*

The goal of this work was to make a tool for modelling what was made under the base of the BPEL specification.

Since the whole process is modelled in the browser of the user, it was our care to make all the process considering this fact, as well as to analyse the graph and, if possible, create the xml output file.

Each shape represents one element in the BPEL specification (with the exception of some shapes). This model was used to offer a graphic representation for users that are used to modelled BPEL processes, and are familiarized with its elements and rules.

We tried, as much as possible, to have some dependencies between the stencil set description file (JSON file) and the BPEL generator plug-in. The stencil set is a file that describes all the shapes, set their properties and roles, establishing role dependencies.

The dependencies were achieved defining the “*id*” of each shape, which normally represents the element, its properties “*id*” is the field attribute or the inner element and the “*optional*” field describes whether the property is optional in BPEL specification. After setting the dependencies and the whole stencil set, the plug-in made as aim of this work analyses the graph created by Oryx, distinguishes the shapes and their properties and generates the *xml* file. This file is created thanks to the use of nodes (26) with a defined “*namespace*” created by a predefined function “*CreateElementNS*” of the object “*Document*” (26) of the browser (every browser that support *javascript* implement it). In this case, it will be an empty string, because the name space is not known yet, and we want to put the proper namespace that is given by the property of the BPEL shape. By the use of functions available in the Node object we are able to set father-child relationship in order to create a tree of elements and the setting of custom parameters and their value.

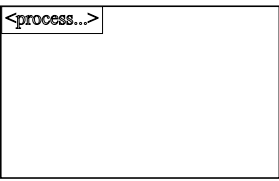
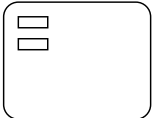
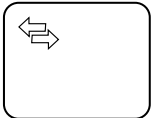
Once you create the whole tree, and set all the attributes and so on, we use the *XMLSerializer* (26) object, available on Firefox browser, which generates a string that represents the tree nodes in xml language.

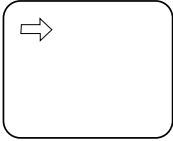






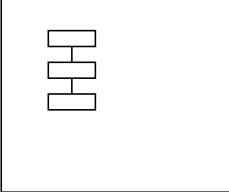
## 5.2 Representing the WS-BPEL

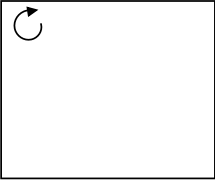

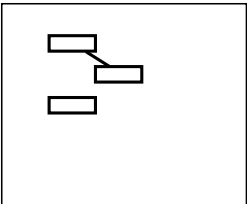
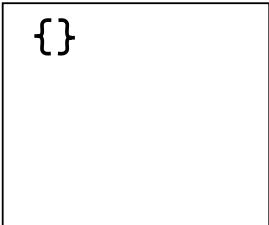




Like it was referred before, the purpose of this project was to describe a graphical representation for the WS-BPEL, creating a block-structured graphical object to represent the same elements in the BPEL specification. Being the editor made to support BPMN models, it was not our strategy to translate a BPMN model to BPEL but instead, having an organization closer to the BPEL elements. Because of this factor, news shapes were made to represent all of the BPEL elements.



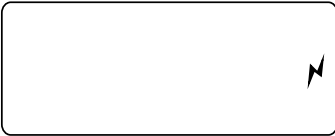

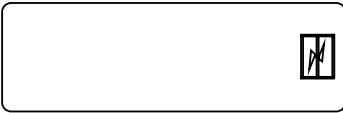
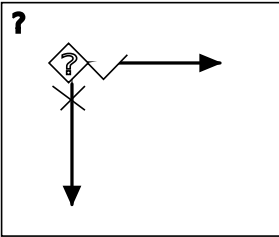


## 5.3 List of the shapes

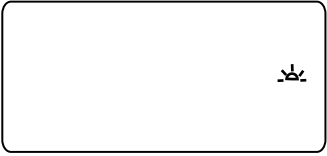



Here is the description of each shape, available in the oryx depository with a brief comment on their features.

Shape and Icon	Description	Comments
<p>&lt;process&gt;</p> 	Represents the sheet layout for a BPEL process	
<p>&lt;assign&gt;</p> 	Updates the values of variables with new data	* data update operations defined as extension under other namespaces is not supported
<p>&lt;invoke&gt;</p> 	Allows the business process to <invoke> a one-way or request-response operation on a portType offered by a partner	

Shape and Icon	Description	Comments
<code>&lt;receive&gt;</code> 	Waits for a matching message to arrive	
<code>&lt;reply&gt;</code> 	Allows the business process to send a message in reply to a message that was received	
<code>&lt;empty&gt;</code> 	"no-op" in a business process	
<code>&lt;opaqueActivity&gt;</code> 	Represents a opaque Activity	
<code>&lt;validate&gt;</code> 	Validates the values of variables against their associated XML and WSDL data definition	
<code>&lt;extensionActiviy&gt;</code> 	Represents an extension Activity for the WS-BPEL by introducing a new activity type	
<code>&lt;wait&gt;</code> 	Waits for a given time period or until a certain point in time has been reached	* the user can choose if is for or until in property <i>ForOrUntil</i> , the definition of these elements is made under the properties <i>expressionlanguage</i> and <i>expression</i>
<code>&lt;sequence&gt;</code> 	Defines a collection of activities to be performed sequentially	

Shape and Icon	Description	Comments
<p>&lt;while&gt;</p> 	The child activity is to be repeated as long as the specified <condition> is true.	
<p>&lt;repeatUntil&gt;</p> 	A cycle activity like while , used to execute the child activity at least once, since the condition is evaluated after the execution	
<p>&lt;flow&gt;</p> 	Specifies one or more activities to be performed concurrently. <links> can be used within a <flow> to define explicit control dependencies between nested child activities.	* to organize the sequence is necessary to use the sequenceFlow shapes ( that links the shapes)
<p>&lt;scope&gt;</p> 	Defines nested activity with its own associated <partnerLinks>, <messageExchanges>, <variables>, <correlationSets>, <faultHandlers>, <compensationHandler>, <terminationHandler>, and <eventHandlers>.	
<p>&lt;compensateScope&gt;</p> 	Compensates on a specified inner scope that has already completed successfully	
<p>&lt;compensate&gt;</p> 	Starts compensation on all inner scopes that have already completed successfully, in default order	
<p>&lt;compensationHandler&gt;</p> 	Compensates in a Scope	
<p>&lt;throw&gt;</p> 	Generates a fault from inside the business process.	

Shape and Icon	Description	Comments
<code>&lt;rethrow&gt;</code> 	Rethrows the fault that was originally caught by the immediately enclosing fault handler	
<code>&lt;exit&gt;</code> 	Immediately end the business process instance	
<code>&lt;faultHandlers&gt;</code> 	With the use of the inner elements <code>&lt;catch&gt;</code> and <code>&lt;catchAll&gt;</code> , it is possible catch the faults caused. This shape can be appended in the <code>&lt;scope&gt;</code> , <code>&lt;process&gt;</code> and <code>&lt;invoke&gt;</code> , space were the faults can be caught	* The use of at least one <code>&lt;catch&gt;</code> or <code>&lt;catchAll&gt;</code> elements should be within this element
<code>&lt;catchAll&gt;</code> 	Catch a fault	* defined in a <code>&lt;faultHandlers&gt;</code> or in a <code>&lt;invoke&gt;</code>
<code>&lt;catch&gt;</code> 	Catches one specific fault	
<code>&lt;if&gt;</code> 	Selects exactly one activity for execution from a set of choices.	* The Shape <i>condition</i> represents the element condition in the <i>if/otherwise</i> element (see section 0 for more details)
<code>&lt;terminationHandler&gt;</code> 	Provides the ability for scopes to control the semantics of forced termination to some degree	
<code>&lt;pick&gt;</code> 	Waits for one of several possible messages to arrive or for a time-out to occur. When one of these triggers occurs, the associated child activity is performed	Contains the inner elements <code>&lt;onAlarm&gt;</code> and <code>&lt;onEventMessage&gt;</code>

Shape and Icon	Description	Comments
<code>&lt;onEvent &gt;</code> 	Indicates that the specified event waits for a message to arrive	
<code>&lt;onAlarm&gt;</code> 	The <code>&lt;onAlarm&gt;</code> corresponds to a timer-based alarm. If the specified duration value in <code>&lt;for&gt;</code> is zero or negative, or a specified deadline in <code>&lt;until&gt;</code> has already been reached or passed, then the <code>&lt;onAlarm&gt;</code> event is executed immediately	* the user can choose if is for or until in property <i>ForOrUntil</i> , the definition of these elements is made under the properties <i>expressionlanguage</i> and <i>expression</i>
<code>&lt;eventhandlers&gt;</code> 	Represents a <code>&lt;eventHandlers&gt;</code> element	Contains the inner elements <code>&lt;onAlarm&gt;</code> and <code>&lt;onMessage&gt;</code>
<code>&lt;onMessage&gt;</code> 	The <code>&lt;onMessage&gt;</code> is similar to a <code>&lt;receive&gt;</code> activity, in that it waits for the receipt of an inbound message	

## 5.4 Restriction with the use of roles

In the stencil set specification there is a set of rules from each element. These are the kind of children's that can be appended as well as their amount.

In order to achieve that, it was used the settings of roles, in the stencil set specification, to restrict the actions for the user. There are some roles shared among the several shapes, but, since we want to use a common role, it was necessary to overcome a limitation of the Oryx editor (described in shape design chapter) that did not supported this feature. Some restrictions have been introduced with the type of elements that can be appended in each element, as well as the amount of inner elements. For instance, only certain shape can contain other shapes.



For instance: One *eventHandlers* element can only exist once in the *process* element as well as in the *scope* element.

In a *scope* there can only be one *Activity* whatsoever; in a flow element you can have multiple *Activities*, so we defined two kinds of roles, one that holds only one *Activity* and the other one that supports multiple *Activities*, being both of these roles given to the shapes that have an *Activity* behaviour.

In the table below is represented the relation between the roles,

Role	Can Contain Shape with Role
PoolHolder	Several Activities
ActivityHolder	One Activity
CatchHolder	Several Catch and/or one CatchAll
If	Condition
eventHandlers	Several OnEvent, onAlarm
pick	OnMessage, onAlarm
HandlerHolder	FaultHandlers, terminationHandler, eventHandlers
compensationHandlerHolder	compensationHandler

*Table 1: Description of roles*

## 5.5 Overcoming some limitations of the Editor

Like was referred before here we will describe some methods used to overcome some limitations of the editor.

### 5.5.1 Order of Execution

As referred before, there is no specific order in a shapes child, but for instance, in the elements like *ifelse*, and *sequence* a specific order of execution is needed. Because of this factor, the use of a link to describe the sequence was, in our point of view, the best solution founded.

#### Using *SequenceOrder* edges to describe a Sequence

In order to know the first Shape in a Sequence, an analysis is made in order to detect which is the child, the one that has no incoming sequence Order links. Because of this factor, it is absolutely necessary that all the childs in a sequence are connected. A restriction to use the Sequence Order was made. With the help of the callback function “canConnect” in stencilset, it was possible to define a restriction, being now only possible to connect two Activities that are childs of a sequence Shape.

### 5.5.2 Presenting the WS-BPEL XML

The BPEL XML string is created within a javascript in the Browser client. Because of the fact that the browser has no privileges to create or delete files, a new way of presenting the xml has to be found.

Two alternatives are possible:

- \* Using a new tab for avoid the server
- \* Using a php page to retry the xml or a set of xml files (in a zip file)

Let's see a little more in detail, how the both work.

Launching the plug-in in mode “avoidServer” with the value *true*, that is available in the *Properties* frame of the canvas, a new tab is opened and the content of the xml is displayed.

On the contrary, if the mode “avoidServer” is set to *false* or more than one process is drawn in the canvas, a post to a php site is made, and the single xml or a zip file containing the xml of several processes is retrieved. In the latter case the name file will be the same as the canvas and the names of each file in it will be the names of the respective processes.

The first one is preferable since it avoids communication to the server, being all created locally. The withdraw is that this is not the standard behaviour, and it cannot be assured that it will work even with future versions of Firefox (on which this plug-in was designed).

### 5.5.3 Creation of new Fields

#### 5.5.3.1 Complex Field

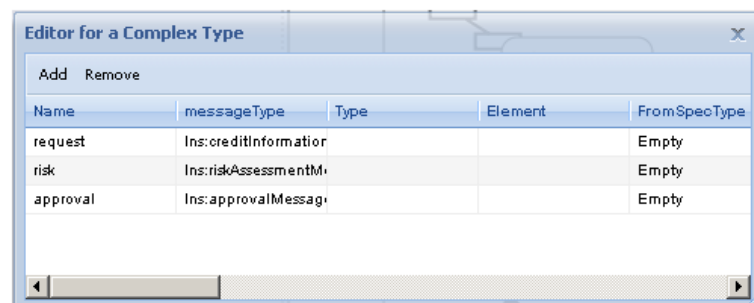


Figure 5-1: Editor of Complex Type

A “*complex*” field (23) is used to describe and to set the properties of an element that has more than one feature.

To keep the generic definition of a stencil set, the complex property type has to be defined in the JSON (9) file.

In the editor, complex properties can be entered by a dialog that contains a table with the property values. After being closed, this dialog will generate a JSON string that holds the several values of the dialog box.

Form the previous version, some changes has to be made in order to be able to create the searcher field, which will be described in the next paragraph.

In particular, few lines have been added to the code in correspondence of the complex field definition.

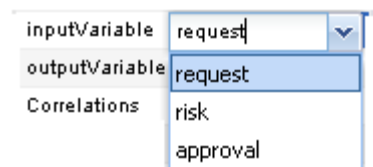
```
else if( type== ORYX.CONFIG.TYPE_SEARCHER)
{
editor = new Ext.grid.GridEditor(
    new Ext.form.varCombo(this.currentShape,this.items[i],parent);
);
}
```

### 5.5.3.2 Searcher Field

In order to create a more dynamic interaction between the several shapes and their properties, it was our interest to create a new kind of field that is composed of definition of the properties of their parent shapes.

For this reason, it had been necessary to create a new type of field as well as to set some definitions in the JSON file.

The Figure 5-I and Figure 5-II describe the interaction between the searcher field in a `<invoke>` shape, and the complex field in the process that contains the name of the variables defined.



*Figure 5-II: Search Field*

```
{
  "id": "variable",
  "prefix": "oryx",
  "type": "searcher",
  "searchBPELId": "BPEL", "searchScopeId": "Scope",
  "searchPropertieBPEL": "variables", "searchPropertieScope": "variables",
  "title": "inputVariable",
  "value": "",
  "description": "",
  "tooltip": "",
  "readonly": false,
  "optional": true,
  "length": undefined
}
```

### 5.5.4 Properties and Shapes

In this chapter we will make the description of some shapes as well as showing important details, in particular their properties that can represent either an attribute or an inner element (or a set of inner Elements).

Among the several properties composing a shape, there are some of those that represent the attributes of the respective element. Once each task has xml elements, we chose to represent some of these “inner” elements as properties. As an example, we will describe the shapes “*Invoke*”, “*Flow*” and “*If*”. Some important properties will also be described for their complexity.

#### 5.5.4.1 Standard Elements and Standard Attributes

The BPEL specification defines some attributes and inner elements as “standard” for an Activity, this field will be given to the several Shapes that have an Activity role.

The Attributes are the “name” and the “*suppressJoinFailure*” that are optional, this last one can have the value “yes” and “no”. The “Standard-Elements” are the “targets” and “sources” needed to describe the dependencies of the Activity in a Flow. The *joinCondition* element in the target Element is necessary to be described in the shapes properties, since is shared between the several incoming links, this field only will be analysed in the occurrence of an incoming link. This element is described with the properties “*JoinCond\_expLang*” and “*JoinCond\_boolExp*” that represents the *expressionLanguage* and the boolean Expression respectively.

The property documentation is related to an optional element where can be added some documentation of the related shape.

The property *backgroundColour* that can be found in several shapes is related to the graphical representation of the shape, being by this way possible to change the background colour of the shape. This property will be ignored in the generation of the BPEL XML.

### 5.5.4.2 Invoke

The *invoke* shape represents an `<invoke>` element. In the repository of shapes, it can be found under the group *Actions*.



It is an activity that allows the business process to `<invoke>` a one-way or request-response operation on a `portType` offered by a partner. In the request-response case, the `<invoke>` activity completes when the response is received. The `portType` attribute on the `<invoke>` activity is optional. If the `portType` attribute is included for readability, the value of the `portType` attribute **MUST** match the `portType` value implied by the combination of the specified `partnerLink` and the `role` implicitly specified by the activity.

The `<invoke>` can also hold a *catch*, *catchAll* and *compensationHandler*. Because of that, it is/was necessary to give the roles *CatchHolder* and *compensationHolder* besides the *Activity* role and the *PoolChild*, both representative of the fact that `<invoke>` is an Activity.

In the Figure 5-IV we can see a catch being appended within the *invoke* activity.

In the stencil Set, the lines that describe this shape are described in Appendix A.1.

Properties >>	
Name	Value
name	assessor
documentation	
JoinCond_expL	
JoinCond_booll	
suppressJoinFai	fromParent
partnerLink	assessor
portType	Ins:riskAssessmentP
operation	check
inputVariable	request
outputVariable	risk
Correlations	
toParts	
fromParts	
BackgroundColor	#ffffff

Figure 5-III: Properties frame of Invoke

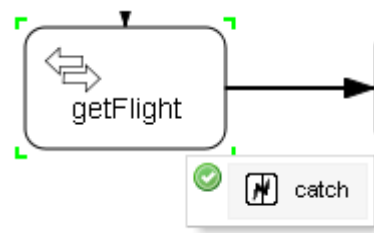


Figure 5-IV: catch handler in Invoke Activity

The Figure 5-III is extracted from the example Loan and describes the properties available for *invoke*.

Some of those properties are standard for tasks and they were already described in the Section 5.5.4.1.

The other properties not described earlier are the following:

### **PartnerLink**

This property is a searcher field, and represents the mandatory attribute with the same name.

Because of this, the property from the type *searcher* will analyse all the enclosed scopes and at least the last element the BPEL process for the property “*partnerLinks*”, processing his complex value (JSON string) and showing as option, the several elements founded.

The value represents the partner which offers the operation.

### **InputVariable and outputVariable**

Both of these properties represent a non-mandatory attribute with the same name.

Like the previous property, also these properties are from the type *searcher*, but in both of these cases the searched property will be the “*variables*”.

One-way invocation requires only the `inputVariable` (or its equivalent `<toPart>` elements) since a response is not expected as part of the operation.

Request-response invocation requires both an `inputVariable` (or its equivalent `<toPart>` elements) and an `outputVariable` (or its equivalent `<fromPart>` elements).

### **Operation and portType**

Both of these properties are text fields and representing the attributes with the same name. Being the *operation* a mandatory character and the *portType* property field an optional.

Being the value the operation offered by a partner has a mandatory to be filled.



## Correlations

The correlation is a complex field composed with the properties: *Correlation*, *initiate* and *pattern*. The *Correlation* is a *searcher* type and the remaining properties are of the *Choice* type.

When the operation invoked is a request/response operation, a pattern attribute on the `<correlation>` specification is used to indicate whether the correlation applies to the outbound message (“**request**”), the inbound message (“**response**”), or both (“**request-response**”). The **pattern** attribute used in `<invoke>` is required for request-response operations, and disallowed when a one-way operation is invoked.

## toParts and fromParts

The `toParts` and the `fromParts` fields are complex fields and optional.

Within the property *toParts* it is possible to create multiple `<toPart>` elements. This element is composed by a *part* attribute (represented in a text field with the same name), and a *correlation* name (represented in searcher field, that loads all the correlations names).

The property *fromParts*, if defined, represents multiple `<fromPart>` elements, which are composed with a *part* attribute (represented in a text field with the same name), and a *toVariable* name (represented in searcher field, that loads all the variable names).

The `<fromPart>` element is used to retrieve data from an incoming multi-part WSDL message and place it into individual WS-BPEL variables. When a WSDL message is received on an `<invoke>` activity that uses `<fromPart>` elements, the message is placed in an anonymous temporary WSDL variable of the type specified by the relevant WSDL operation's output message. The `<fromPart>` elements, as a group, act as a single virtual `<assign>`, with each `<fromPart>` acting as a `<copy>`.

### 5.5.4.3 If

This activity consists of an ordered list of one or more conditional branches defined by the `<if>` and optional `<elseif>` elements, followed by an optional `<else>` element. The `<if>` and `<elseif>` branches are considered in the order in which they appear. The first branch whose `<condition>` holds true is taken, and its contained activity is performed. If no branch with a condition is taken, then the `<else>` branch is taken if present.

Like describe before in the chapter 5.5.1 there was the use of another shapes to fully support the `<if>` element.

The `<if>` element is composed by the following shapes: an *If holder* shape that represents the main activity, at least one *if condition* shape has to be present, representing the `<condition>` element, making it possible to connect the conditions with two kinds of links, the *ifTrue* if the condition is true and a *ifFalse* if the same fails.

All of these shapes can be found under the group “Controls if”.

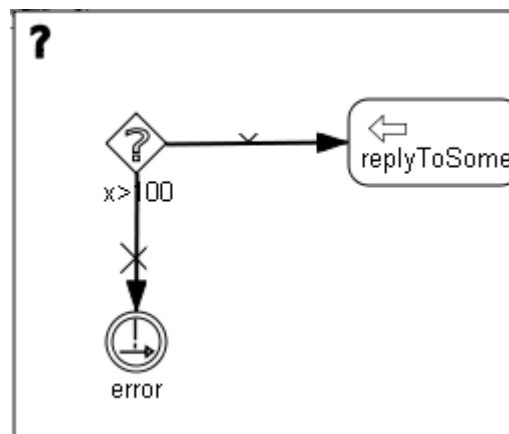
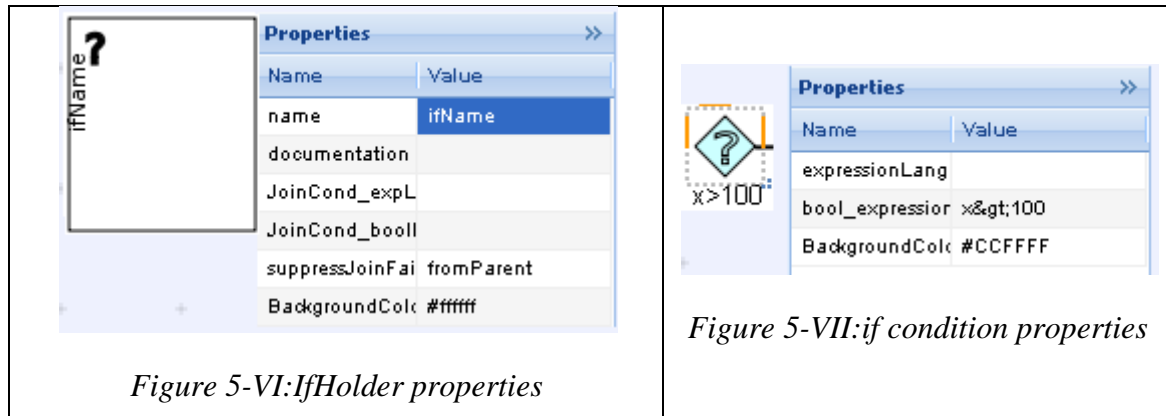


Figure 5-V:if example

In the example above (Figure 5-V) is possible to notice the pictures to which we referred before.

This example is the demonstration of a condition that analyses the `x` variable: if it is greater than ‘100’ it will be executed the “*replyToSome*” activity or else a throw “*error*” will be executed.

More than one condition could be used, appended in the false link of the condition, to produce an *elseif* sequence. If no *IfFalse* link is supplied to the *if condition* shape, the `<else>` element will not be produced.



Like stated before, the *condition* shape represents a `<condition>` element in the `<if>` element, or in `<elseif>`, and is described by the properties that can be seen in the Figure 5-VII, since the *bool\_expression* is a mandatory character, being a condition always needed.

The expression *language* property represents the *expressionLanguage* attribute of the referred element. Although optional, if it does not hold any value, it will not be presented in the xml of the process. The *If holder* shape contains only the standard attributes, as well as the standard elements, being this ones described in the section 5.5.4.1.

In the stencil Set the lines that describe this shape are described in Appendix A.2.

#### 5.5.4.4 Flow

The `<flow>` activity provides concurrency and synchronization.

A fundamental semantic effect of grouping a set of activities in a `<flow>` is to enable concurrency. A `<flow>` ends when all of the activities enclosed by the `<flow>` have been completed.

A `<flow>` activity creates a set of concurrent activities directly nested in it. It enables synchronization dependencies between activities that are nested in it to any depth. The `<link>` construct is used to express these synchronization dependencies.

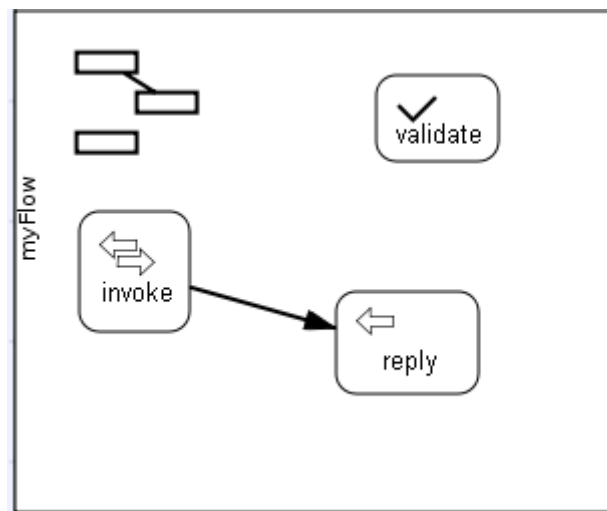


Figure 5-VIII: Flow Example

In Figure 5-VIII is described an example of a *flow* activity.

This example consists in a reply shape dependent to the invoke shape, and the *validate* shape that can be executed concurrently; all of this shapes are nested in the *flow* activity.

In this section is described the *Flow* node and the *sequenceFlow* edge that represents the `<flow>` element and the *link* elements. The *flow* shape has the standard elements and standard attributes as we can see in the Figure 5-X.

The `<links>` composed in the `<flow>` element are specified after the analysis of the children.

Properties >>	
Name	Value
Name	FlowLink
documentation	
expression	>100
expressionLang	

Figure 5-IX:sequenceFlow properties

Properties >>	
Name	Value
name	myFlow
documentation	
JoinCond_expL	
JoinCond_boolI	
suppressJoinFai	fromParent
BackgroundColo	#ffffff

Figure 5-X:Flow properties

The Figure 5-IX shows the properties related with the *sequenceLink*.

This edge is composed by the properties *Name* - that correspond to the *linkName* attribute of the `<link>` element, an *expression* that represents the Boolean expression - and an *expressionLanguage* - an attribute with the same name, being the last two optional and part of the `<transitionCondition>` element nested inside of the `<link>` available in the `<source>` elements of the standard attributes.

The descriptions in the *stencilset* file of these shapes can be found in the Appendix A.3.

## 6 Examples

Some standard demos have been reproduced in order to show some potentialities and possible applications of the WS-BPEL.

There will be shown two cases, “Loan” and “Travel”

In reference [ ] it is used a WSDL file that is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information.

We predispose the plug-in for the use of this file, in the “import” field, but nevertheless we decided not to load it, instead, directly choosing the shapes and setting the parameters into the editor, letting its implementation as a part of a future work.

### 6.1 Loan Approval

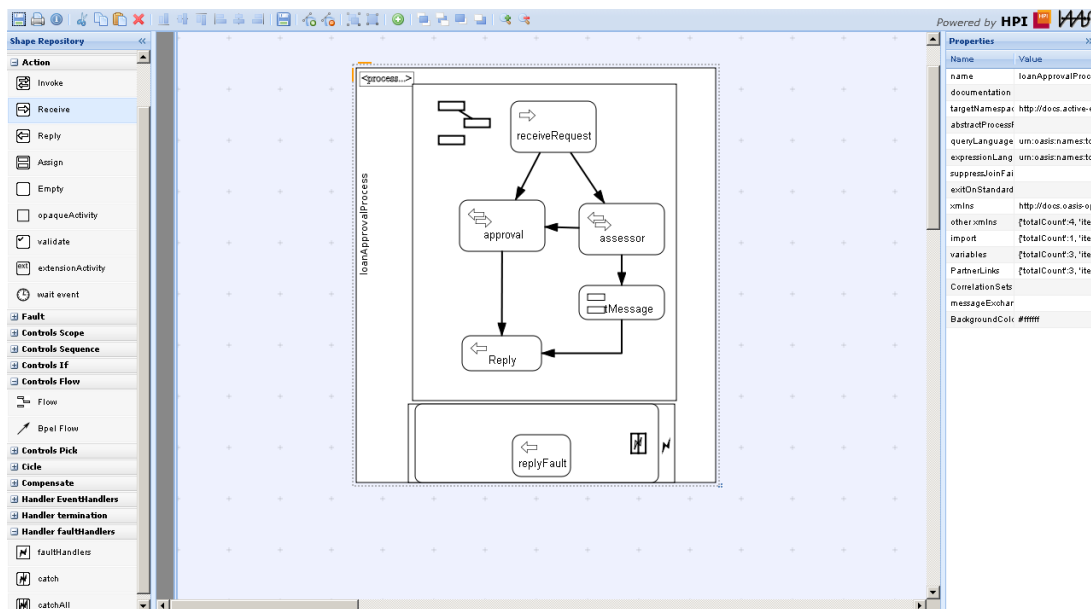


Figure 6-1: Loan Approval Demo (31)

This example consists of a simple loan approval service. Customers of the service send loan requests, including personal information and amount being requested. Using this information, the loan service executes a simple process.

In the process, the interaction with the customer is represented by the initial `<receive>` and the matching `<reply>` activities. The use of risk assessment and loan approval services is represented by `<invoke>` elements. All these activities are contained within a `<flow>`, and their (potentially concurrent) behavior is executed according to the dependencies expressed by the `<link>` elements. Note that the transition conditions attached to the `<source>` elements of the links determine which links get activated. Dead path elimination is enabled by setting the `suppressJoinFailure` attribute to `yes` on the `<process>` element.

The operations invoked can return a fault of type `loanProcessFault`, therefore a fault handler is provided. When a fault occurs, control is transferred to the fault handler where a `<reply>` element is used to return a fault response of type `unableToHandleRequest` to the loan requester.

The XML file generated corresponds to the one used in the reference to create this example; it can be seen in chapter Appendix B.1.

In the Appendix C.1 we can find the value that describes the persistence of the data on the server, which represents the shapes, their properties and their relations.

## 6.2 Travel

This process describes the flow of a travel agency.

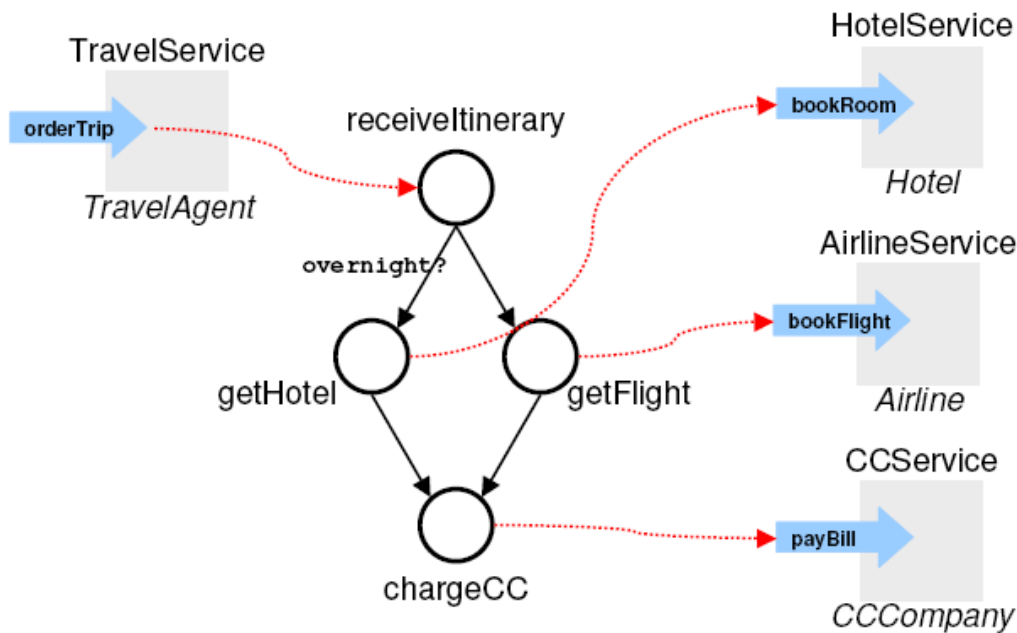


Figure 6-II: Travel Flow Process

The user insert the search parameter for a trip, then the Web service generates a list of possibilities, among which the user will choose his destination or search for other possibilities.

When he decides to choose one of the form presented, he will be asked to provide the data of his credit card, and the receipt is generated and added to the travel plan. Whether the trip involves one night to spend in a hotel, the operation serves this possibility or it is evaluated if it is necessary to book a flight.

Both the activities *getHotel* and *getFlight* calls respectively the WebService for the hotel with the operation *bookRoom*, the AirlineService, through the operation *bookFlight* and deliver the few parameters in the message of the operation.

In the message *flightBooking* of the AirlineService the following parameters are given: number of persons, departure airport, destination airport, dates of departure and return flight. The AirlineService make the flight book according to the data submitted. Same way it works the Room booking.



The last activity is the “*chargeCC*” that allow the operation *payBill* on the WebService, that concludes the process.

In the following figure, can be seen how this process have been represented in Oryx

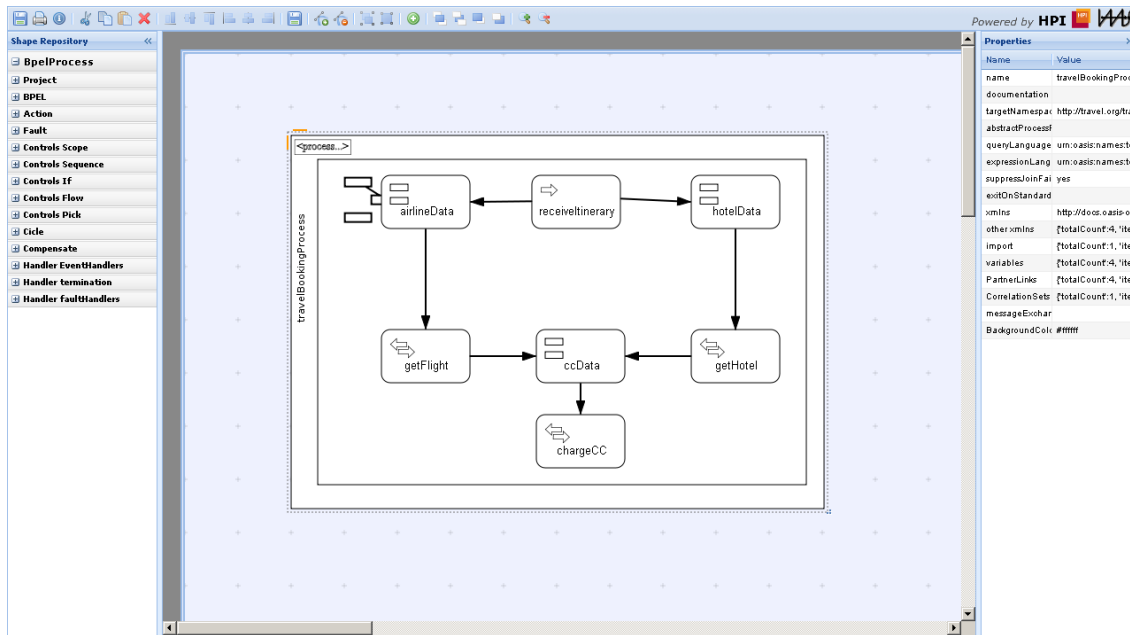


Figure 6-III: Travel process

In the Appendix B.2 we can see the XML file generated by the plug-in

## 7 Conclusions

In this report has been presented the results obtained thanks to the creation of a plug-in for the Web oriented editor Oryx, that was aimed to generate an xml file containing the description of a work flow made using the language of Web Services Business Process Execution Language (WS-BPEL). It was not used the BPMN strategy but we decided instead to create a block-structured graphical object to represent the same elements in the BPEL specification, so, an hypothetical user who does not know the BPEL specification as well as its rules and its inner elements, should first be acquainted to this language in order to be able to model a work flow. Through the creation of new *javascript* file, a new field has been introduced, the *searcher*, which was not available before. Many relationships among the shapes, not existing before, were created, in particular thanks to the new *searcher* field, as well as through a different use of the roles.

In order to limit the user actions to a set of valid one, in a logical sense, some constraints had to be put. For instance, now the control *Scope* can contain just one *Activity*.

Despite the fact that the editor is largely extensible and flexible, nevertheless, some bugs have been found and have already been discussed throughout this paper, but there are still some bugs in the editor that are still waiting to be solved.

## 8 Future work

In this Chapter some of the future possible developments of this work have been represented, thinking in possible integrations to the plug-in and to the stencil set.

### 8.1 *Creating the missing elements*

One shape is missing; it is the “*forEach*” shape, which corresponds to the `<foreach>` element described in the WS-BPEL specification. Its function is to execute an activity a certain number of times specified in one of his attribute field.

### 8.2 *Loading definition files*

One of the next steps of development should be loading and analysing the files described in the `<import>` elements, for instance an external XML Schema or WSDL definitions. WSDL files define services as collections of network endpoints, or ports.

Instead of being the user to fill each field, the idea is to load all data, analyse it and show as multiple choice.

The advantage is to avoid mistakes in modelling, creating an interface more user-friendly.

### 8.3 *BPEL4Chor*

Another future development could be giving full support to BPEL4Chor since at the moment, only the BPEL architecture is supported.

## **8.4 Validation of the process**

Not all of the xml files generated are valid.

A possible validation of the process should be done, to avoid the generation of an incorrect BPEL. As a valid BPEL we meant that all the necessary elements should be described. For instance a `<flow>` element should contain at least one Activity, but since is the user to drag and drop the activities is not guarantee that an activity exists being necessary to validate the xml produced.

This validation can be made with the use of an XML schema (21) or with the use of the XPath (19) queries.

At the moment no schema validation is available in the browser, leaving only the XPath queries, or a validation in the server.

## **8.5 Loading an external BPEL**

An external BPEL created, or not, within the Oryx editor has also to be loaded on it.

A BPEL process itself does not contain any graphical representation. Because of it, the loading of an external BPEL file can be complex. For this reason, we propose loading to be restricted with the files produced by the Editor. Since it is possible to save the data, that is the graphical representation (size, location, etc), and to hold that same definitions under other namespace, in the xml. In this way is possible to load and analyse it, in order to create the shapes.

## 9 Appendix

### Appendix A.1

The following code describes the definitions of an *invoke* shape, with his properties, roles and callback functions.

```
{
  "type": "node",
  "id": "invoke",
  "title": "Invoke",
  "groups": ["Action"],
  "description": "An invoke activity.",
  "view": "action/node.action.invoke.svg",
  "icon": "new_invoke.png",
  "roles": ["Activity", "PoolChild", "CatchHolder", "compensationHandlerHolder"],
  "properties": [
    {
      "id": "name",
      "prefix": "oryx",
      "type": "String",
      "title": "name",
      "value": "",
      "description": "",
      "tooltip": "",
      "readonly": false,
      "optional": true,
      "refToView": "label",
      "length": undefined,
      "wrapLines": false
    },
    {
      "id": "documentation",
      "prefix": "oryx",
      "type": "String",
      "title": "documentation",
      "value": "",
      "description": "",
      "tooltip": "",
      "readonly": false,
      "optional": true,
      "refToView": "",
      "length": undefined,
      "wrapLines": true
    },
    {
      "id": "JC_expLang",
```

```
"prefix":"oryx",
"type":"String",
"title":"JoinCond_expLang",
"value":"",
"description":"",
"tooltip":"",
"readonly":false,
"optional":true,
"refToView":"",
"length":undefined,
"wrapLines":false
},
{
  "id":"JoinCond_boolExp",
  "prefix":"oryx",
  "type":"String",
  "title":"JoinCond_boolExp",
  "value":"",
  "description":"",
  "tooltip":"",
  "readonly":false,
  "optional":true,
  "refToView":"",
  "length":undefined,
  "wrapLines":false
},
{
  "id":"suppressJoinFailure",
  "type":"Choice",
  "title":"suppressJoinFailure",
  "value":"fromParent",
  "description":"",
  "tooltip":"",
  "readonly":false,
  "optional":true,
  "refToView":"",
  "items": [
    {
      "id":"fromParent",
      "title":"fromParent",
      "value":"fromParent",
      "tooltip":"",
      "refToView":"none"
    },
    {
      "id":"yes",
      "title":"yes",
      "value":"yes",
      "tooltip":"",
```

```
        "refToView": "none"
    },
    {
        "id": "no",
        "title": "no",
        "value": "no",
        "tooltip": "",
        "refToView": "none"
    }
]
},
{
    "id": "partnerLink",
    "prefix": "oryx",
    "type": "searcher",
    "searchBPELId": "BPEL", "searchScopeId": "Scope",
    "searchPropertieBPEL": "PartnerLinks", "searchPropertieScope": "PartnerLinks",
    "title": "partnerLink",
    "value": "",
    "description": "",
    "tooltip": "",
    "readonly": false,
    "optional": false,
    "length": undefined
},
{
    "id": "portType",
    "prefix": "oryx",
    "type": "String",
    "title": "portType",
    "value": "",
    "description": "",
    "tooltip": "",
    "readonly": false,
    "optional": true,
    "refToView": "",
    "length": undefined,
    "wrapLines": false
},
{
    "id": "operation",
    "prefix": "oryx",
    "type": "String",
    "title": "operation",
    "value": "",
    "description": "",
    "tooltip": "",
    "readonly": false,
    "optional": false,
```

```
"refToView":"","  
  "length":undefined,  
  "wrapLines":false  
},  
{  
  "id":"inputVariable",  
  "prefix":"oryx",  
  "type":"searcher",  
  "searchBPELId":"BPEL","searchScopeId":"Scope",  
  "searchPropertieBPEL":"variables","searchPropertieScope":"variables",  
  "title":"inputVariable",  
  "value":"","  
  "description":"","  
  "tooltip":"","  
  "readonly":false,  
  "optional":true,  
  "length":undefined  
},  
{  
  "id":"outputVariable",  
  "prefix":"oryx",  
  "type":"searcher",  
  "searchBPELId":"BPEL","searchScopeId":"Scope",  
  "searchPropertieBPEL":"variables","searchPropertieScope":"variables",  
  "title":"outputVariable",  
  "value":"","  
  "description":"","  
  "tooltip":"","  
  "readonly":false,  
  "optional":true,  
  "length":undefined  
},  
{  
  "id":"Correlations",  
  "type":"Complex",  
  "title":"Correlations",  
  "value":"","  
  "description":"","  
  "readonly":false,  
  "optional":true,  
  "complexItems": [  
    {  
      "id":"Correlation",  
      "name":"Correlation",  
      "type":"searcher",  
      "searchBPELId":"BPEL","searchScopeId":"Scope",  
      "searchPropertieBPEL":"CorrelationSets","searchPropertieScope":"CorrelationSets",  
      "width":100,  
      "value":"","
```



```
"optional":true,
"prefix":"oryx"
},
{
  "id":"initiate",
  "name":"initiate",
  "type":"Choice",
  "value":"Empty",
  "width":100,
  "optional":true,
  "items": [
    {
      "id":"yes",
      "title":"yes",
      "value":"yes",
      "refToView":""
    },
    {
      "id":"join",
      "title":"join",
      "value":"join",
      "refToView":""
    },
    {
      "id":"no",
      "title":"no",
      "value":"no",
      "refToView":""
    }
  ]
},
{
  "id":"pattern",
  "name":"pattern",
  "type":"Choice",
  "value":"Empty",
  "width":100,
  "optional":true,
  "items": [
    {
      "id":"request",
      "title":"request",
      "value":"request",
      "refToView":""
    },
    {
      "id":"response",
      "title":"response",
      "value":"response",
```

```

        "refToView":"response"
    },
    {
        "id":"request_response",
        "title":"request_response",
        "value":"request_response",
        "refToView":""
    }
]
}
],
{
    "id":"toParts",
    "type":"Complex",
    "title":"toParts",
    "value":"",
    "description":"",
    "readonly":false,
    "optional":true,
    "complexItems": [
        {
            "id":"part",
            "name":"part",
            "type":"String",
            "value":"",
            "width":100,
            "optional":false
        },
        {
            "id":"Correlation",
            "name":"Correlation",
            "type":"searcher",
            "searchBPELId":"BPEL", "searchScopeId":"Scope",
            "searchPropertieBPEL":"variables", "searchPropertieScope":"variables",
            "width":100,
            "value":"",
            "optional":false,
            "prefix":"oryx"
        }
    ]
},
{
    "id":"fromParts",
    "type":"Complex",
    "title":"fromParts",
    "value":"",
    "description":"",
    "readonly":false,

```

```
"optional":true,
"complexItems": [
  {
    "id":"part",
    "name":"part",
    "type":"String",
    "value":"",
    "width":100,
    "optional":false
  },
  {
    "id":"toVariable",
    "name":"toVariable",
    "type":"searcher",
    "searchBPELId":"BPEL","searchScopeId":"Scope",
    "searchPropertieBPEL":"variables","searchPropertieScope":"variables",
    "width":100,
    "value":"",
    "optional":false,
    "prefix":"oryx"
  }
]
},
{
  "id":"bgColor",
  "type":"Color",
  "title":"BackgroundColor",
  "value":"#ffffff",
  "description":"",
  "readonly":false,
  "optional":false,
  "refToView":"mainBody",
  "fill":true,
  "stroke":false
}
]
},
```

## Appendix A.2

### If Holder

The *If Holder* shape is here presented with his properties. This shape represents the main shape of the element `<if>`. The Standard elements and standard properties of the referred element are here described .

```
{
  "type": "node",
  "id": "if",
  "title": "If Holder",
  "description": "One If Holder",
  "groups": ["Controls If"],
  "view": "controls/controls.ifHolder.svg",
  "icon": "new_ifHolder.png",
  "roles": ["PoolHolder", "Activity", "PoolChild", "If"],
  "properties": [{
    "id": "name",
    "prefix": "oryx",
    "type": "String",
    "title": "name",
    "value": "",
    "description": "",
    "tooltip": "",
    "readonly": false,
    "optional": true,
    "refToView": "label",
    "length": undefined,
    "wrapLines": false
  },
  {
    "id": "documentation",
    "prefix": "oryx",
    "type": "String",
    "title": "documentation",
    "value": "",
    "description": "",
    "tooltip": "",
    "readonly": false,
    "optional": true,
    "refToView": "",
    "length": undefined,
    "wrapLines": true
  },
  {
    "id": "JC_expLang",
    "prefix": "oryx",
    "type": "String",
    "title": "JoinCond_expLang",
    "value": "",
    "description": "",
    "tooltip": "",
    "readonly": false,
    "optional": true,
    "refToView": "",
    "length": undefined,
    "wrapLines": false
  },
  {
    "id": "JoinCond_boolExp",
    "prefix": "oryx",
    "type": "String",
    "title": "JoinCond_boolExp",
    "value": "",
    "description": "",
    "tooltip": "",
    "readonly": false,
  }
  ]
}
```

```

        "optional":true,
        "refToView":"","
        "length":undefined,
        "wrapLines":false
    },
    {
        "id":"suppressJoinFailure",
        "type":"Choice",
        "title":"suppressJoinFailure",
        "value":"fromParent",
        "description":"","
        "tooltip":"","
        "readonly":false,
        "optional":true,
        "refToView":"","
        "items": [
            {
                "id":"fromParent",
                "title":"fromParent",
                "value":"fromParent",
                "tooltip":"","
                "refToView":"none"
            },
            {
                "id":"yes",
                "title":"yes",
                "value":"yes",
                "tooltip":"","
                "refToView":"none"
            },
            {
                "id":"no",
                "title":"no",
                "value":"no",
                "tooltip":"","
                "refToView":"none"
            }
        ]
    },
    {
        "id":"bgColor",
        "type":"Color",
        "title":"BackgroundColor",
        "value":"#ffffff",
        "description":"","
        "readonly":false,
        "optional":false,
        "refToView":"mainBody",
        "fill":true,
        "stroke":false
    }
],
},

```

## If condition

The shape *if\_condition* represents a `<condition>` nested inside a `<if>` element.

```
{
  "type": "node",
  "id": "if_condition",
  "title": "If condition",
  "groups": ["Controls If"],
  "description": "If condition",
  "view": "controls/node.if_condition.svg",
  "icon": "new_if_condition.png",
  "roles": ["Condition"],
  "properties": [
    {
      "id": "expressionLanguage",
      "type": "String",
      "title": "expressionLanguage",
      "value": "",
      "description": "",
      "readonly": false,
      "optional": true,
      "length": "50",
      "wrapLines": false
    },
    {
      "id": "bool_expression",
      "type": "String",
      "title": "bool_expression",
      "value": "",
      "description": "",
      "refToView": "label",
      "readonly": false,
      "optional": false,
      "length": "30"
    },
    {
      "id": "bgColor",
      "type": "Color",
      "title": "BackgroundColor",
      "value": "#ffffff",
      "description": "",
      "readonly": false,
      "optional": false,
      "refToView": "mainBody",
      "fill": true,
      "stroke": false
    }
  ]
},
```

## If True link

This edge shape defines a valid dependency between the *if\_condition* and an Activity.

```
{
  "type": "edge",
  "id": "sequenceIfTrue",
  "title": "If True",
  "description": "Connects the Activity related if the condition is True",
  "groups": ["Controls If"],
  "view": "controls/edge.ifTrue.svg",
  "icon": "new_sequence_flow.png",
  "roles": ["sequenceIf", "if"],
  "properties": []
},
```

## If False link

With this link it is possible to define an Activity or another condition.

```
{
  "type": "edge",
  "id": "sequenceIfFalse",
  "title": "If False",
  "description": "Defines the Activity related within the condition if is True",
  "groups": ["Controls If"],
  "view": "controls/edge.ifFalse.svg",
  "icon": "new_sequence_flow.png",
  "roles": ["Controls Sequence", "if"],
  "properties": []
}
```

## Appendix A.3

In this appendix it is possible to see the JSON code responsible for the creation of the shape *Flow* and its properties

```
{
  "type": "node",
  "id": "flow",
  "title": "Flow",
  "description": "One Flow",
  "groups": ["Controls Flow"],
  "view": "controls/controls.flow.svg",
  "icon": "new_flow.png",
  "roles": ["PoolHolder", "Activity", "PoolChild", "Flow"],
  "properties": [
    {
      "id": "name",
      "prefix": "oryx",
      "type": "String",
      "title": "name",
      "value": "",
      "description": "",
      "tooltip": "",
      "readonly": false,
      "optional": true,
      "refToView": "label",
      "length": undefined,
      "wrapLines": false
    },
    {
      "id": "documentation",
      "prefix": "oryx",
      "type": "String",
      "title": "documentation",
      "value": "",
      "description": "",
      "tooltip": "",
      "readonly": false,
      "optional": true,
      "refToView": "",
      "length": undefined,
      "wrapLines": true
    },
    {
      "id": "JC_expLang",
      "prefix": "oryx",
      "type": "String",
      "title": "JoinCond_expLang",
      "value": "",
      "description": "",
      "tooltip": "",
      "readonly": false,
      "optional": true,
      "refToView": "",
      "length": undefined,
      "wrapLines": false
    },
    {
      "id": "JoinCond_boolExp",
      "prefix": "oryx",
      "type": "String",
      "title": "JoinCond_boolExp",
      "value": "",
      "description": "",
      "tooltip": "",
      "readonly": false,
      "optional": true,
      "refToView": "",
      "length": undefined,

```



```

        "wrapLines":false
    },
    {
        "id":"suppressJoinFailure",
        "type":"Choice",
        "title":"suppressJoinFailure",
        "value":"fromParent",
        "description":"",
        "tooltip":"",
        "readonly":false,
        "optional":true,
        "refToView":"",
        "items": [
            {
                "id":"fromParent",
                "title":"fromParent",
                "value":"fromParent",
                "tooltip":"",
                "refToView":"none"
            },
            {
                "id":"yes",
                "title":"yes",
                "value":"yes",
                "tooltip":"",
                "refToView":"none"
            },
            {
                "id":"no",
                "title":"no",
                "value":"no",
                "tooltip":"",
                "refToView":"none"
            }
        ]
    },
    {
        "id":"bgColor",
        "type":"Color",
        "title":"BackgroundColor",
        "value":"#ffffff",
        "description":"",
        "readonly":false,
        "optional":false,
        "refToView":"mainBody",
        "fill":true,
        "stroke":false
    }
],
},

```

## Appendix B.1

In the code showed below it is possible to analyse the BPEL created by the BPELGenerator plug-in. This XML<sup>1</sup> is referred to the Loan Approval presented in the chapter 6.1

```
<process name="loanApprovalProcess" targetNamespace="http://docs.active-
endpoints.com/loanApproval.bpel" abstractProcessProfile=""
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"
expressionLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"
xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:lns="http://docs.active-endpoints.com/loanApproval.wsdl"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <import namespace="http://example.com/loan-approval/wsdl/"
location="loanServicePT.wsdl" importType="http://schemas.xmlsoap.org/wsdl/" />
  <variables>
    <variable name="request" messageType="lns:creditInformationMessage" />
    <variable name="risk" messageType="lns:riskAssessmentMessage" />
    <variable name="approval" messageType="lns:approvalMessage" />
  </variables>
  <PartnerLinks>
    <PartnerLink name="customer" PartnetLinkType="lns:loanPartnerLinkType"
myRole="loanService" />
    <PartnerLink name="approver" PartnetLinkType="lns:loanApprovalLT"
partnerRole="approver" />
    <PartnerLink name="assessor" PartnetLinkType="lns:riskAssessmentLT"
partnerRole="assessor" />
  </PartnerLinks>
  <faultHandlers>
    <catch inputVariable="error" faultMessageType="lns:errorMessage"
faultElement="lns:loanProcessFault">
      <reply name="replyFault" partnerLink="customer"
portType="lns:loanServicePT" operation="request" variable="error"
faultName="unableToHandleRequest" />
    </catch>
  </faultHandlers>
  <flow name="mainFlow">
    <links>
      <link linkName="receive-to-approval" />
      <link linkName="receive-to-assess" />
      <link linkName="assess-to-approval" />
      <link linkName="approval-to-reply" />
      <link linkName="assess-to-setMessage" />
      <link linkName="setMessage-to-reply" />
    </links>
    <receive name="receiveRequest" partnerLink="customer"
portType="lns:loanServicePT" operation="request" variable="request">
      <sources>
        <source linkName="receive-to-approval">
          <transitionCondition>$request.amount &gt;=
50000</transitionCondition>
        </source>
        <source linkName="receive-to-assess">
```

<sup>1</sup> The XML presented here was formatted, with tabs and paragraphs, since the string that represents it had none of it.

```

        <transitionCondition>$request.amount <lt;
50000</transitionCondition>
    </source>
</sources>
</receive>
    <invoke name="approval" partnerLink="approver"
portType="lns:loanApprovalPT" operation="approve" inputVariable="request"
outputVariable="approval">
    <targets>
        <target linkName="assess-to-approval"/>
        <target linkName="receive-to-approval"/>
    </targets>
    <sources>
        <source linkName="approval-to-reply"/>
    </sources>
</invoke>
    <invoke name="assessor" partnerLink="assessor"
portType="lns:riskAssessmentPT" operation="check" inputVariable="request"
outputVariable="risk">
    <targets>
        <target linkName="receive-to-assess"/>
    </targets>
    <sources>
        <source linkName="assess-to-approval">
            <transitionCondition>$risk.level!='low'</transitionCondition>
        </source>
        <source linkName="assess-to-setMessage">
            <transitionCondition>$risk.level='low'</transitionCondition>
        </source>
    </sources>
</invoke>
    <assign name="setMessage">
        <copy>
            <from>
                <literal>yes</literal>
            </from>
            <to variable="approval" part="accept"/>
        </copy>
        <targets>
            <target linkName="assess-to-setMessage"/>
        </targets>
        <sources>
            <source linkName="setMessage-to-reply"/>
        </sources>
    </assign>
    <reply name="Reply" partnerLink="customer" portType="lns:loanServicePT"
operation="request" variable="approval">
        <targets>
            <target linkName="approval-to-reply"/>
            <target linkName="setMessage-to-reply"/>
        </targets>
    </reply>
</flow>
</process>

```

## Appendix B.2

In the code showed bellow is possible to analyse the BPEL created by the BPELGenerator plug-in. This XML<sup>2</sup> is referred to the Loan Approval presented in the chapter 6.1

The XML here presented was formatted, with tabs and paragraphs, since the string that represents it had none of it.

```
<process name="travelBookingProcess"
targetNamespace="http://travel.org/travelBooking.bpel"
abstractProcessProfile=""
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"
expressionLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"
xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:tag="http://travel.org/travelBooking.wsdl">
  <import namespace="http://travel.org/travelBooking.wsdl"
location="project:TravelBookingSzenario/travelBooking.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/">
  <variables>
    <variable name="Order" messageType="tag:trip"/>
    <variable name="Hotel" messageType="tag:hotelBooking"/>
    <variable name="Flight" messageType="tag:flightBooking"/>
    <variable name="Payment" messageType="tag:payments"/>
  </variables>
  <PartnerLinks>
    <PartnerLink name="Customer" PartnetLinkType="tag:Traveler"
myRole="TravelAgent"/>
    <PartnerLink name="Hotel" PartnetLinkType="tag:Hotel"
myRole="HotelChain"/>
    <PartnerLink name="Airline" PartnetLinkType="tag:Airline"
myRole="AirlineCompany"/>
    <PartnerLink name="Billing" PartnetLinkType="tag:CreditCard"
myRole="CreditCardCompany"/>
  </PartnerLinks>
  <CorrelationSets>
    <CorrelationSet name="correlation" properties="ns3:piid"/>
  </CorrelationSets>
  <flow>
    <links>
      <link linkName="Itinerary-to-Flight"/>
      <link linkName="Itinerary-to-Hotel"/>
      <link linkName="to-Hotel"/>
      <link linkName="Hotel-to-Charge"/>
      <link linkName="to-Flight"/>
      <link linkName="Flight-to-Charge"/>
      <link linkName="to-Charge"/>
    </links>
    <receive name="receiveItinerary" partnerLink="Customer"
portType="tag:TravelService" operation="orderTrip" variable="Order">
      <sources>
        <source linkName="Itinerary-to-Flight"/>
        <source linkName="Itinerary-to-Hotel"/>
      </sources>
    </receive>
  </flow>
</process>
```

<sup>2</sup> The XML presented here was formatted, with tabs and paragraphs, since the string that represents it had none of it.

```

        <transitionCondition>overnight=???true???</transitionCondition>
    </source>
</sources>
</receive>
<assign name="hotelData">
    <copy>
        <from>$Order.itinerary/rooms</from>
        <to/>
    </copy>
    <targets>
        <target linkName="Itinerary-to-Hotel"/>
    </targets>
    <sources>
        <source linkName="to-Hotel"/>
    </sources>
</assign>
<invoke name="getHotel" partnerLink="Hotel" portType="tag:HotelService"
operation="bookRoom" inputVariable="Hotel">
    <targets>
        <target linkName="to-Hotel"/>
    </targets>
    <sources>
        <source linkName="Hotel-to-Charge"/>
    </sources>
</invoke>
<assign name="airlineData">
    <copy>
        <from>$Order.itinerary/flights</from>
        <to/>
    </copy>
    <targets>
        <target linkName="Itinerary-to-Flight"/>
    </targets>
    <sources>
        <source linkName="to-Flight"/>
    </sources>
</assign>
<invoke name="getFlight" partnerLink="Airline"
portType="tag:AirlineService" operation="bookFlight" inputVariable="Flight">
    <targets>
        <target linkName="to-Flight"/>
    </targets>
    <sources>
        <source linkName="Flight-to-Charge"/>
    </sources>
</invoke>
<invoke name="chargeCC" partnerLink="Billing"
portType="tag:CreditCardService" operation="payBill" inputVariable="Payment">
    <targets>
        <target linkName="to-Charge"/>
    </targets>
</invoke>
<assign name="ccData">
    <copy>
        <from>$Order.customerInfo</from>
        <to/>
    </copy>
    <copy>
        <from>$Hotel</from>
        <to/>
    </copy>
    <copy>
        <from>$Flight</from>
        <to/>
    </copy>

```

```
<targets>
  <joinCondition>$Hotel-to-Charge or $Flight-to-Charge</joinCondition>
  <target linkName="Flight-to-Charge"/>
  <target linkName="Hotel-to-Charge"/>
</targets>
<sources>
  <source linkName="to-Charge"/>
</sources>
</assign>
</flow>
</process>
```

## Appendix C.1

The elements<sup>3</sup> shown below represent the saved state of all shapes and their properties in the Server, each `<span>` element describes the property, since the attribute `'class'` holds its name.

Each shape is represented by the element `<div>`, and its dependencies with the element `<a>`. The referred HXTML belongs to the loanApproval example described in the chapter 6.1.

```
<div id="resource0">
  <span class="oryx-type">http://b3mn.org/stencilset/b3mn#BPEL</span>
  <br/>
  <span class="oryx-name">loanApprovalProcess</span>
  <br/>
  <span class="oryx-documentation"></span>
  <br/>
  <span class="oryx-targetNamespace">http://docs.active-
endpoints.com/loanApproval.bpel</span>
  <br/>
  <span class="oryx-abstractProcessProfile"></span>
  <br/>
  <span class="oryx-
queryLanguage">urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0</span>
  <br/>
  <span class="oryx-
expressionLanguage">urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0</span>
  <br/>
  <span class="oryx-suppressJoinFailure"></span>
  <br/>
  <span class="oryx-exitOnStandardFault"></span>
  <br/>
  <span class="oryx-xmlns">http://docs.oasis-
open.org/wsbpel/2.0/process/executable</span>
  <br/>
  <span class="oryx-otherxmlns">{'totalCount':4, 'items':[{prefix:"bpel",
namespace:"http://docs.oasis-open.org/wsbpel/2.0/process/executable\""},
{prefix:"bpws", namespace:"http://schemas.xmlsoap.org/ws/2003/03/business-
process/\""}, {prefix:"lns", namespace:"http://docs.active-
endpoints.com/loanApproval.wsdl\""}, {prefix:"xsd",
namespace:"http://www.w3.org/2001/XMLSchema\""}]}</span>
  <br/>
  <span class="oryx-import">{'totalCount':1,
'items':[{namespace:"http://example.com/loan-approval/wsdl/",
location:"loanServicePT.wsdl",
importType:"http://schemas.xmlsoap.org/wsdl/\""}]}</span>
  <br/>
  <span class="oryx-variables">{'totalCount':3, 'items':[{name:"request",
messageType:"lns:creditInformationMessage", type:"", element:"",
fromspecType:"Empty", fromspecvariablename:"", fromspecpart:"",
fromspecpartnerLink:"", fromspecendpointReference:"",
fromspecqueryLanguage:"", fromspecquery:"", fromspecproperty:"",
fromspecexpressionLanguage:"", fromspecexpression:"", fromspecliteral:""},
{name:"risk", messageType:"lns:riskAssessmentMessage", type:"", element:"",
fromspecType:"Empty", fromspecvariablename:"", fromspecpart:"",
fromspecpartnerLink:"", fromspecendpointReference:"",
fromspecqueryLanguage:"", fromspecquery:"", fromspecproperty:""},
```

<sup>3</sup> The elements presented here were formatted, with tabs and paragraphs, since the string that represents it had none of it.

```

fromspecexpressionlanguage:"", fromspecexpression:"", fromspecliteral:""},
{name:"approval", messageType:"lns:approvalMessage", type:"", element:"",
fromspectype:"Empty", fromspecvariablename:"", fromspecpart:"",
fromspecpartnerLink:"", fromspecendpointReference:"",
fromspecquerylanguage:"", fromspecquery:"", fromspecproperty:"",
fromspecexpressionlanguage:"", fromspecexpression:"",
fromspecliteral:""}]]</span>
<br/>
<span class="oryx-PartnerLinks">{'totalCount':3, 'items':[{name:"customer",
PartnetLinkType:"lns:loanPartnerLinkType", myRole:"loanService",
partnerRole:"", initializePartnerRole:""}, {name:"approver",
PartnetLinkType:"lns:loanApprovalLT", myRole:"", partnerRole:"approver",
initializePartnerRole:""}, {name:"assessor",
PartnetLinkType:"lns:riskAssessmentLT", myRole:"", partnerRole:"assessor",
initializePartnerRole:""}]]</span>
<br/>
<span class="oryx-CorrelationSets"></span>
<br/>
<span class="oryx-messageExchanges"></span>
<br/>
<span class="oryx-bgColor">#ffffff</span>
<br/>
<span class="oryx-bounds">209,95.03125,630,581.03125</span>
<br/>
</div>
<div id="resource9">
<span class="oryx-type">http://b3mn.org/stencilset/b3mn#faultHandlers</span>
<br/>
<span class="oryx-documentation"></span>
<br/>
<span class="oryx-bgColor">#ffffff</span>
<br/>
<span class="oryx-bounds">61,394.00000381469727,373,486</span>
<br/>
<a rel="raziel-parent" href="#resource0"/>
</div>
<div id="resource10">
<span class="oryx-type">http://b3mn.org/stencilset/b3mn#catch</span>
<br/>
<span class="oryx-faultName"></span>
<br/>
<span class="oryx-inputVariable">error</span>
<br/>
<span class="oryx-documentation"></span>
<br/>
<span class="oryx-faultMessageType">lns:errorMessage</span>
<br/>
<span class="oryx-faultElement">lns:loanProcessFault</span>
<br/>
<span class="oryx-bgColor">#ffffff</span>
<br/>
<span class="oryx-bounds">7,14.968746185302734,224,91.99999618530273</span>
<br/>
<a rel="raziel-parent" href="#resource9"/>
</div>
<div id="resource2">
<span class="oryx-type">http://b3mn.org/stencilset/b3mn#reply</span>
<br/>
<span class="oryx-name">replyFault</span>
<br/>
<span class="oryx-documentation"></span>
<br/>
<span class="oryx-JC_expLang"></span>
<br/>
<span class="oryx-JoinCond_boolExp"></span>

```



```

<br/>
<span class="oryx-suppressJoinFailure">fromParent</span>
<br/>
<span class="oryx-partnerLink">customer</span>
<br/>
<span class="oryx-portType">lns:loanServicePT</span>
<br/>
<span class="oryx-operation">request</span>
<br/>
<span class="oryx-variable">error</span>
<br/>
<span class="oryx-faultName">unableToHandleRequest</span>
<br/>
<span class="oryx-messageExchange"></span>
<br/>
<span class="oryx-Correlations"></span>
<br/>
<span class="oryx-toParts"></span>
<br/>
<span class="oryx-bgColor">#ffffff</span>
<br/>
<span class="oryx-bounds">56,18,124,59.062503814697266</span>
<br/>
<a rel="raziel-parent" href="#resource10"/>
</div>
<div id="resource7">
  <span class="oryx-type">http://b3mn.org/stencilset/b3mn#flow</span>
  <br/>
  <span class="oryx-name">mainFlow</span>
  <br/>
  <span class="oryx-documentation"></span>
  <br/>
  <span class="oryx-JC_expLang"></span>
  <br/>
  <span class="oryx-JoinCond_boolExp"></span>
  <br/>
  <span class="oryx-suppressJoinFailure">fromParent</span>
  <br/>
  <span class="oryx-bgColor">#ffffff</span>
  <br/>
  <span class="oryx-bounds">66,18.96875,375,389.96875</span>
  <br/>
  <a rel="raziel-parent" href="#resource0"/>
</div>
<div id="resource5">
  <span class="oryx-type">http://b3mn.org/stencilset/b3mn#receive</span>
  <br/>
  <span class="oryx-name">receiveRequest</span>
  <br/>
  <span class="oryx-documentation"></span>
  <br/>
  <span class="oryx-JC_expLang"></span>
  <br/>
  <span class="oryx-JoinCond_boolExp"></span>
  <br/>
  <span class="oryx-suppressJoinFailure">fromParent</span>
  <br/>
  <span class="oryx-partnerLink">customer</span>
  <br/>
  <span class="oryx-portType">lns:loanServicePT</span>
  <br/>
  <span class="oryx-operation">request</span>
  <br/>
  <span class="oryx-variable">request</span>
  <br/>

```

```

<span class="oryx-messageExchange"></span>
<br/>
<span class="oryx-createInstance">yes</span>
<br/>
<span class="oryx-Correlations"></span>
<br/>
<span class="oryx-bgColor">#ffffff</span>
<br/>
<span class="oryx-
bounds">115,19.999996185302734,215,79.99999618530273</span>
<br/>
<a rel="raziel-outgoing" href="#resource14"/>
<a rel="raziel-outgoing" href="#resource8"/>
<a rel="raziel-parent" href="#resource7"/>
</div>
<div id="resource4">
  <span class="oryx-type">http://b3mn.org/stencilset/b3mn#invoke</span>
  <br/>
  <span class="oryx-name">approval</span>
  <br/>
  <span class="oryx-documentation"></span>
  <br/>
  <span class="oryx-JC_expLang"></span>
  <br/>
  <span class="oryx-JoinCond_boolExp"></span>
  <br/>
  <span class="oryx-suppressJoinFailure">fromParent</span>
  <br/>
  <span class="oryx-partnerLink">approver</span>
  <br/>
  <span class="oryx-portType">lns:loanApprovalPT</span>
  <br/>
  <span class="oryx-operation">approve</span>
  <br/>
  <span class="oryx-inputVariable">request</span>
  <br/>
  <span class="oryx-outputVariable">approval</span>
  <br/>
  <span class="oryx-Correlations"></span>
  <br/>
  <span class="oryx-toParts"></span>
  <br/>
  <span class="oryx-fromParts"></span>
  <br/>
  <span class="oryx-bgColor">#ffffff</span>
  <br/>
  <span class="oryx-bounds">55,136,155,196</span>
  <br/>
  <a rel="raziel-outgoing" href="#resource12"/>
  <a rel="raziel-parent" href="#resource7"/>
</div>
<div id="resource3">
  <span class="oryx-type">http://b3mn.org/stencilset/b3mn#invoke</span>
  <br/>
  <span class="oryx-name">assessor</span>
  <br/>
  <span class="oryx-documentation"></span>
  <br/>
  <span class="oryx-JC_expLang"></span>
  <br/>
  <span class="oryx-JoinCond_boolExp"></span>
  <br/>
  <span class="oryx-suppressJoinFailure">fromParent</span>
  <br/>
  <span class="oryx-partnerLink">assessor</span>

```

```

<br/>
<span class="oryx-portType">lns:riskAssessmentPT</span>
<br/>
<span class="oryx-operation">check</span>
<br/>
<span class="oryx-inputVariable">request</span>
<br/>
<span class="oryx-outputVariable">risk</span>
<br/>
<span class="oryx-Correlations"></span>
<br/>
<span class="oryx-toParts"></span>
<br/>
<span class="oryx-fromParts"></span>
<br/>
<span class="oryx-bgColor">#ffffff</span>
<br/>
<span class="oryx-
bounds">195,139.99999618530273,295,199.99999618530273</span>
<br/>
<a rel="raziel-outgoing" href="#resource11"/>
<a rel="raziel-outgoing" href="#resource13"/>
<a rel="raziel-parent" href="#resource7"/>
</div>
<div id="resource6">
<span class="oryx-type">http://b3mn.org/stencilset/b3mn#assign</span>
<br/>
<span class="oryx-name">setMessage</span>
<br/>
<span class="oryx-documentation"></span>
<br/>
<span class="oryx-JC_expLang"></span>
<br/>
<span class="oryx-JoinCond_boolExp">assessment-to-accept or ($approval-to-
accept and $approvalSecure-to-accept) or($approval-to-accept and
not($loanRequest-to-accept))</span>
<br/>
<span class="oryx-suppressJoinFailure">fromParent</span>
<br/>
<span class="oryx-validate"></span>
<br/>
<span class="oryx-copy">{'totalCount':1, 'items':[{keepSrcElementName:"",
ignoreMissingFromData:"fromParent", fromspectype:"Literal",
fromspecvariablename:"", fromspecpart:"", fromspecpartnerLink:"",
fromspecendpointReference:"", fromspecquerylanguage:"", fromspecquery:"",
fromspecproperty:"", fromspecexpressionlanguage:"", fromspecexpression:"",
fromspecliteral:"yes", tospectype:"Variable", tospecvariablename:"approval",
tospecpart:"accept", tospecpartnerLink:"", tospecquerylanguage:"",
tospecquery:"", tospecproperty:"", tospecexpressionlanguage:"",
tospecexpression:""}]}</span>
<br/>
<span class="oryx-bgColor">#ffffff</span>
<br/>
<span class="oryx-bounds">195,236,295,276</span>
<br/>
<a rel="raziel-outgoing" href="#resource15"/>
<a rel="raziel-parent" href="#resource7"/>
</div>
<div id="resource1">
<span class="oryx-type">http://b3mn.org/stencilset/b3mn#reply</span>
<br/>
<span class="oryx-name">Reply</span>
<br/>
<span class="oryx-documentation"></span>
<br/>

```

```

<span class="oryx-JC_expLang"></span>
<br/>
<span class="oryx-JoinCond_boolExp"></span>
<br/>
<span class="oryx-suppressJoinFailure">fromParent</span>
<br/>
<span class="oryx-partnerLink">customer</span>
<br/>
<span class="oryx-portType">lns:loanServicePT</span>
<br/>
<span class="oryx-operation">request</span>
<br/>
<span class="oryx-variable">approval</span>
<br/>
<span class="oryx-faultName"></span>
<br/>
<span class="oryx-messageExchange"></span>
<br/>
<span class="oryx-Correlations"></span>
<br/>
<span class="oryx-toParts"></span>
<br/>
<span class="oryx-bgColor">#ffffff</span>
<br/>
<span class="oryx-bounds">58,295,151,336</span>
<br/>
<a rel="raziel-parent" href="#resource7"/>
</div>
<div id="resource11">
  <span class="oryx-type">http://b3mn.org/stencilset/b3mn#sequenceFlow</span>
  <br/>
  <span class="oryx-linkName">assess-to-approval</span>
  <br/>
  <span class="oryx-documentation"></span>
  <br/>
  <span class="oryx-transition_expression">$risk.level!='low'</span>
  <br/>
  <span class="oryx-TC_expressionLanguage"></span>
  <br/>
  <span class="oryx-textrotation"></span>
  <br/>
  <span class="oryx-
bounds">469.5667274206366,282.5590469143757,469.7816579127618,282.565187779722
86</span>
  <br/>
  <a rel="raziel-outgoing" href="#resource4"/>
  <span class="oryx-dockers">50 30 50 30 # </span>
  <br/>
  <a rel="raziel-parent" href="#resource7"/>
</div>
<div id="resource14">
  <span class="oryx-type">http://b3mn.org/stencilset/b3mn#sequenceFlow</span>
  <br/>
  <span class="oryx-linkName">receive-to-approval</span>
  <br/>
  <span class="oryx-documentation"></span>
  <br/>
  <span class="oryx-transition_expression">$request.amount >= 50000</span>
  <br/>
  <span class="oryx-TC_expressionLanguage"></span>
  <br/>
  <span class="oryx-textrotation"></span>
  <br/>

```

```

    <span class="oryx-
bounds">439.2689447549416,194.32791388085977,439.7401314491499,249.31768172479
95</span>
    <br/>
    <a rel="raziel-outgoing" href="#resource4"/>
    <span class="oryx-dockers">50 60 50 0 # </span>
    <br/>
    <a rel="raziel-parent" href="#resource7"/>
</div>
<div id="resource8">
    <span class="oryx-type">http://b3mn.org/stencilset/b3mn#sequenceFlow</span>
    <br/>
    <span class="oryx-linkName">receive-to-assess</span>
    <br/>
    <span class="oryx-documentation"></span>
    <br/>
    <span class="oryx-transition_expression">
        $request.amount < 50000</span>
    <br/>
    <span class="oryx-TC_expressionLanguage"></span>
    <br/>
    <span class="oryx-textrotation"></span>
    <br/>
    <span class="oryx-
bounds">466.6506156742804,194.08288812066468,519.2,253.3999961853027</span>
    <br/>
    <a rel="raziel-outgoing" href="#resource3"/>
    <span class="oryx-dockers">50 60 50 0 # </span>
    <br/>
    <a rel="raziel-parent" href="#resource7"/>
</div>
<div id="resource12">
    <span class="oryx-type">http://b3mn.org/stencilset/b3mn#sequenceFlow</span>
    <br/>
    <span class="oryx-linkName">approval-to-reply</span>
    <br/>
    <span class="oryx-documentation"></span>
    <br/>
    <span class="oryx-transition_expression"></span>
    <br/>
    <span class="oryx-TC_expressionLanguage"></span>
    <br/>
    <span class="oryx-textrotation"></span>
    <br/>
    <span class="oryx-
bounds">379.9763319115569,310.1460278615069,379.9944537181978,408.644536842731
67</span>
    <br/>
    <a rel="raziel-outgoing" href="#resource1"/>
    <span class="oryx-dockers">50 30 46.5 20.5 # </span>
    <br/>
    <a rel="raziel-parent" href="#resource7"/>
</div>
<div id="resource13">
    <span class="oryx-type">http://b3mn.org/stencilset/b3mn#sequenceFlow</span>
    <br/>
    <span class="oryx-linkName">assess-to-setMessage</span>
    <br/>
    <span class="oryx-documentation"></span>
    <br/>
    <span class="oryx-transition_expression">$risk.level='low'</span>
    <br/>
    <span class="oryx-TC_expressionLanguage"></span>
    <br/>
    <span class="oryx-textrotation"></span>

```

```

    <br/>
    <span class="oryx-
bounds">520.468386756071,314.68664863295976,521,349.51562413573265</span>
    <br/>
    <a rel="raziel-outgoing" href="#resource6"/>
    <span class="oryx-dockers">50 30 50 20 # </span>
    <br/>
    <a rel="raziel-parent" href="#resource7"/>
</div>
<div id="resource15">
    <span class="oryx-type">http://b3mn.org/stencilset/b3mn#sequenceFlow</span>
    <br/>
    <span class="oryx-linkName">setMessage-to-reply</span>
    <br/>
    <span class="oryx-documentation"></span>
    <br/>
    <span class="oryx-transition_expression"></span>
    <br/>
    <span class="oryx-TC_expressionLanguage"></span>
    <br/>
    <span class="oryx-textrotation"></span>
    <br/>
    <span class="oryx-
bounds">426.52239401096824,390.37824192546344,520,429.5</span>
    <br/>
    <a rel="raziel-outgoing" href="#resource1"/>
    <span class="oryx-dockers">50 20 520 429.5 46.5 20.5 # </span>
    <br/>
    <a rel="raziel-parent" href="#resource7"/>
</div>
<div id="oryxcanvas" class="-oryx-canvas">
    <span class="oryx-mode">writeable</span>
    <span class="oryx-mode">fullscreen</span>
    <a rel="oryx-stencilset" href="./stencilsets/bpel/bpelmodel.json"/>
    <a rel="oryx-render" href="#resource0"/>
    <a rel="oryx-render" href="#resource9"/>
    <a rel="oryx-render" href="#resource10"/>
    <a rel="oryx-render" href="#resource2"/>
    <a rel="oryx-render" href="#resource7"/>
    <a rel="oryx-render" href="#resource5"/>
    <a rel="oryx-render" href="#resource4"/>
    <a rel="oryx-render" href="#resource3"/>
    <a rel="oryx-render" href="#resource6"/>
    <a rel="oryx-render" href="#resource1"/>
    <a rel="oryx-render" href="#resource11"/>
    <a rel="oryx-render" href="#resource14"/>
    <a rel="oryx-render" href="#resource8"/>
    <a rel="oryx-render" href="#resource12"/>
    <a rel="oryx-render" href="#resource13"/>
    <a rel="oryx-render" href="#resource15"/>
</div>

```

## 10 Bibliography

1. **Crusson, Tanguy.** Business Process Management Essentials. *GLiNTECH*. [Online] 2006.  
<http://www.glintech.com/downloads/BPM%20Essentials%20with%20Open%20Source.pdf>.
2. *Web Services Business Process Execution Language Version 2.0.* **OASIS**. 2007.
3. **W3C.** Scalable Vector Graphics (SVG) 1.1 Specification. *W3C*. [Online] 2003.  
<http://www.w3.org/TR/SVG11/REC-SVG11-20030114.pdf>.
4. Introducing JSON. *json.org*. [Online] <http://json.org/>.
5. **Javascript, Extended.** Ext Documentation Center. *extjs*. [Online] 2007.  
<http://extjs.com/deploy/ext-1.1.1/docs/>.
6. **W3C.** XML Path Language (XPath) Version 1.0. *W3.org*. [Online] 199.  
<http://www.w3.org/TR/xpath>.
7. —. XML Schema. *W3.org*. [Online] 2000. <http://www.w3.org/XML/Schema>.
8. —. Web Services Description Language (WSDL) 1.1. *W3C*. [Online] 2001.  
<http://www.w3.org/TR/wsdl>.
9. **Oasis.** wsbpel specification working draft. *Oasis*. [Online] <http://www.oasis-open.org/committees/download.php/10347/wsbpel-specification-draft-120204.htm>.
10. **Tscheschner, Willi.** *Oryx Dokumentation*. Hasso Plattner Institut, Universität Potsdam. Potsdam : s.n., 2007.
11. *Process Definition Capabilities and Web Services.* **Trickovic, Ivana.** 2005, The SAP Developer Network.
12. **White, Stephen A.** Introduction to BPMN. *bpmn.org*. [Online] <http://www.bpmn.org/Documents/Introduction%20to%20BPMN.pdf>.
13. **Team, Prototype Core.** Prototype API Documentation. *Prototype*. [Online] 2007.  
<http://www.prototypejs.org/api>.
14. **Polak, Daniel.** *Oryx: BPMN Stencil Set Implementation*. Hasso Plattner Institut, Universität Potsdam . Potsdam : s.n., 2007.
15. **Pfitzner, Kerstin.** *Choreography Configuration for BPMN*. IAAS, Choreography Configuration for BPMN". Stuttgart : s.n., 2007.
16. **Peters, Nicolas.** *Stencil Set Specification*. Hasso Plattner Institut , Universität Potsdam . Potsdam : s.n., 2007.
17. **Mozilla.** XML Extras. [Online] [developer.mozilla.org](http://developer.mozilla.org/en/docs/XML_Extras) May 2007.  
[http://developer.mozilla.org/en/docs/XML\\_Extras](http://developer.mozilla.org/en/docs/XML_Extras).
18. **center, Mozilla developer.** Document Object Model. *Mozilla developer center*. [Online] September 2007. <http://developer.mozilla.org/en/docs/DOM>.
19. **Moorthy, Kumar Raj.** An Introduction to BPEL. *developer.com*. [Online] 2006.  
<http://www.developer.com/services/print.php/3609381>.
20. *Business processes for Web Services: Principles and applications.* **Leymann, F., Keller, A. and Khalaf, R.** 2006, IBM SYSTEMS JOURNAL, VOL 45, NO 2,.
21. **Czuchra, Martin.** *Oryx: Embedding Business Process Data into the Web*. Hasso Plattner Institut, Universität Potsdam. Potsdam : s.n., 2007.
22. **Wikipedia.** Business process management. *Wikipedia, the free encyclopedia*. [Online] [http://en.wikipedia.org/wiki/Business\\_process\\_management](http://en.wikipedia.org/wiki/Business_process_management).
23. —. BPEL. *Wikipedia*. [Online] 2007. <http://en.wikipedia.org/wiki/BPEL>.

24. **Miguel, Antony.** WS-BPEL 2.0 Tutorial. *Eclipse* . [Online]  
[http://www.eclipse.org/tptp/platform/documents/design/choreography\\_html/tutorials/ws\\_bpel\\_tut.html](http://www.eclipse.org/tptp/platform/documents/design/choreography_html/tutorials/ws_bpel_tut.html).
25. **Bpel, Active.** Creating an Abstract vs. an Executable Process. *Active Bpel*. [Online]  
<http://www.activebpel.org/infocenter/ActiveBPEL/v211/index.jsp?topic=/com.activee.bpel.doc/html/UG7-6.html>.



## **Declaration**

All the work contained within this thesis, except where otherwise acknowledged, was solely the effort of the author. At no stage was any collaboration entered into with any other party.

---

(Bruno Miguel Castanheira Colaço)