Process Modeling Architectures with Namespace and XML Technology

Tiago Lopes Telecken, José Valdeni de Lima
Universidade Federal do Rio Grande do Sul, Instituto de Informática
Porto Alegre – RS, Brazil, Av. Bento Conçalves, 9500 Campus do Vale - Bloco IV
CEP 91591-970, Caixa Postal 15064
{telecken, valdeni} @inf.ufrgs.br

Montgomery Barroso França
Universidade Federal do Rio Grande do Sul, Instituto de Informática

Banco Central do Brasil
Brasília – DF, Brazil, CEP 70074-900, SBS Quadra 3 Bloco B - Ed. Sede
Caixa Postal 08670
mont@bcb.gov.br

Abstract

The necessity of productivity and quality in workflow systems demands the use of several process modeling architectures. However, in the workflow area, there is few information about optional architectures of relationships among models and documents used in the process modeling phase. To attend the demand for information about optional architectures, this paper presents a survey about *many-one* architecture and a comparative study about process modeling architectures. The *many-one* architecture uses namespace and XML technology to insert elements of many XML models in only one process definition. Such characteristic allows a workflow technology development be more modular and reusable.

Keywords: Process Definitions, Process modeling, Namespace, Workflow, XML

1 Introduction

In the document structuring area the most traditional architecture of relationships among models and documents is formed by one document that contains all information used in an application and by one model that defines all structure of referred document. This is the *one-one* architecture.

But, some applications need architectures that contain many models. Two optional architectures were developed for these applications. The first one is formed by many documents. Its respective model defines the structure of each document. This is the *many-many* architecture. The second one is formed by one document, but the structure of this document is defined in many models. This is the *many-one* architecture.

The necessity of productivity and quality in workflow systems and the application of workflow technology in more and more complex environments demand the use of all process modeling architectures.

However, there are few information, developments and researches about architectures of relationships among models and documents used in the process modeling phase. And, this few information is concentrated in the *one-one* architecture. For helping developers and researches to choose the best architecture in process definitions, this paper presents a survey about *many-one* architecture and a comparative study about the process modeling architectures.

2 Overview -Workflow Systems Architecture

In agreement with the Workflow Management Coalition (WfMC) [20][18], the generic architecture of a workflow system should follow the model shown in the figure 1. Concisely this model determines that: the workflow designers can generate a process definition through a definition tool. The process definition should contain (i) all the information that the workflow engine needs to manage, control and execute a workflow; and (ii) all the information that the definition tool needs to facilitate the process definition edition. After, the process definition can be sent to a workflow engine. The workflow engine will interpret the process definition. After, it will control, manage and execute the workflow described in the process definition.

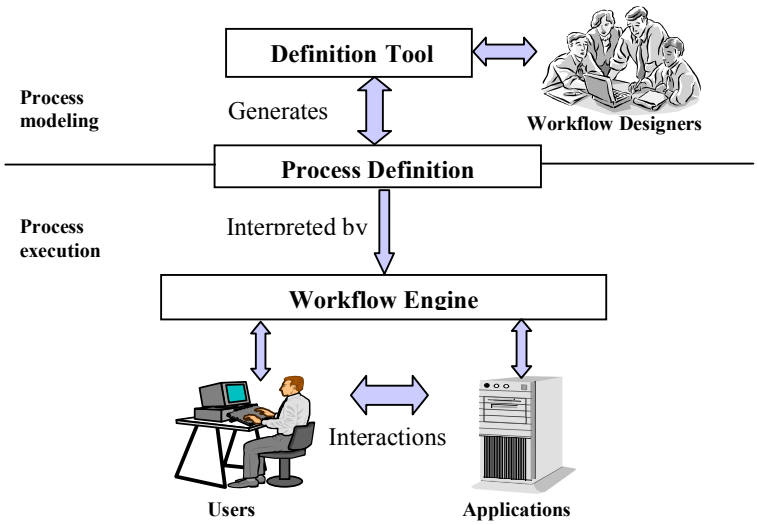

Fig. 1. Workflow Systems

The elements and attributes contained in a process definition are defined in one model called *process definition model*. The process definitions are of two types. The first one is the internal representations. Internal representations were projected to be the internal data representation of a specific definition tool. The second one is the interchange patterns. The interchange patterns allow the compatibility among the systems evolved in a workflow process.

Process definitions are complex documents that have a rigid syntax. So, in the process modeling phase, the use of a definition tool is essential. The referred tool facilitates the analysis, modeling and codification of a process definition.

The main components and functionalities of the definition tool are [15]:

- **Internal representation:** It is a process definition which structure is defined by the process definition model adopted by the definition tool.
- **Internal representation control**: It is the internal representation edition resources.
- **Visions**: They are functionalities that supply the users with one vision of the internal representation. Graphic and textual visions can be supplied.

- **Export/Import process definitions**: Export is the resource that converts the internal representation of a workflow process to an external format. Import is the resource that converts a process definition of an external format to the internal representation.
- **Nesting**: They are functionalities that allow the process definition hierarchy modeling and navigation.
- **Error verification**: It is a functionality that allows the automatic error verification in process definitions. Syntax and semantic errors are verified.
- **Analysis and simulation:** They are functionalities that facilitate the analysis and simulation of modeled workflow behavior.

3 Overview - Using XML in Process Definitions

Extensible Markup Language (XML) technology [4] is a vast and growing set of modules that offer services, tools and standards used in a wide range of areas. Its use in document structuring is largely divulged and offers a growing number of tools.

XML document structuring is largely used in process definition. The XML Process Definition Language (XPDL) [19] is an important XML interchange pattern defined by WfMC (entity created for developing workflow standards). Other important XML interchange patterns are [1]: XLang [17] from Microsoft, Web Services Flow Language (WSFL) [11] from IBM, Business Process Modeling Language (BPML) from Business Process Modeling Language Initiative (BPMI) [2], Web Service Choreography Interface (WSCI) [23] from Sun/BEA and Business Process Execution Language for Web Services (BPEL4WS) [12] (the evolution of XLang and WSFL in the web services context).

Many internal representations of process definitions are XML languages. This is the case of Biztalk Orchestration Designer that uses the XLang [17] and of IBM's MQ Series Workflow that has an internal representation based on WSFL [9].

Most definition tools export and/or import their internal representations for one or several XML formats. The most common exportations and importations are for the following formats: XPDL, XLang, WSFL, BPML, WSCI and BPEL4WS.

The main advantages of XML application in process definitions are:

- A great interoperability with workflow systems (workflow management systems, definition tools, simulation programs, etc). The most important workflow systems are enabled to import, export or interpret XML languages.
- A great interoperability with systems from other areas. XML documents are a standard data exchange format. With XML the process definition and the definition tools become compatible with many tools, protocols, applications and resources such as Extensible Stylesheet Language Transformations (XSLT), XML Schema, Simple API for XML (SAX), Document Object Model (DOM) API, and much more.

The use of XML in process definitions is growing. The main entities and organizations of workflow area already use XML representations. For this reason, the focus of this paper is XML solutions.

3.1 Using Namespace in Process Definition

The specification Namespace [3] is a XML standard that defines how two or more XML representations can be inserted in the same document. This specification is an official recommendation of the World Wide Web Consortium (W3C).

The namespace is used to insert external XML representation into process definitions. This is the case of Resource Description Framework (RDF) elements and attributes inserted in PSL [14]. RDF elements in Process Specification Language (PSL) documents define resources used in workflow systems.

The namespace is also used to put elements of a process definition language in external XML documents. This is the case of XLang elements that are put in Web Services Description Language (WSDL) documents. According to Thatte [17], the WSDL is a net service description protocol. Among other services, the WSDL describes Web Services by XLang elements and attributes.

3.2 Using SVG, XLink, and RDF in Process Definition

The Scalable Vector Graphics (SVG) [22], XML Linking Language (XLink) [6] and RDF [21] specifications are XML vocabularies that describe respectively 2D graphics, links and resources. These languages are official recommendations of World Wide Web Consortium (W3C) and are cited in the cases of this paper. Definition tools such as ILOG [10] export the graphic representation of workflow process to SVG formats. The XLink is used by some definition tools and is proposed by some authors [7] as a good standard to link elements of process definitions localized in different documents.

4 Related works - The Process Modeling Architectures

The approached area of this paper is the architectures of relationships among models and documents used in the process modeling phase. In these architectures, the entities that are relevant for process execution or process modeling are defined in models. The models are applied in domain areas. The entities of a domain that are defined in one model are called entity domains. The entity domains are represented in documents by elements and attributes. The structure of documents used in these architectures is defined in the referred models. There are three possible architectures.

In the first architecture, each process definition is formed by one document. The structure of process definition is defined in only one model. This is the *one-one* architecture. In the current process definition and definition tools, this is the most used architecture. An example of this architecture is XPDL [19]. In this standard, all elements used in the process modeling phase and in the process execution phase are defined in one model and represented in one document.

In the second architecture, each process definition is formed by more than one document. The structure of each document is defined in one different model. The elements of each document may make mutual references. So, the documents that form one process definition need resources to maintain the mutual synchronization. This is the *many-many* architecture. An example is the internal representation of definition tool FORO process designer [8], that is formed by two documents. One document contains elements defined in the process model. The other contains elements defined in the informational model. There are two models and the process definition is formed by two documents.

In the third architecture, each process definition is formed by one document. And the structure of process definition is defined in different models. This is the *many-one* architecture. An example is the PSL [14], an interchange pattern that defines external resources by RDF elements inserted by namespace standard. There are two models, PSL and RDF. The PSL is the main model and RDF is the secondary model.

Each column of table 1 shows the characteristics of one researched architecture and the figure 2 shows the architecture of the three referred examples.

Table 1 - Architecture's characteristics

1- *One-One*	2- *Many-Many*	3- *Many-One*
Process definition structure defined in **one model**	Process definition structure defined in **many models**	Process definition structure defined in **many models**
Process definition is formed by **one document**	Process definition is formed by **many documents**	Process definition is formed by **one document**

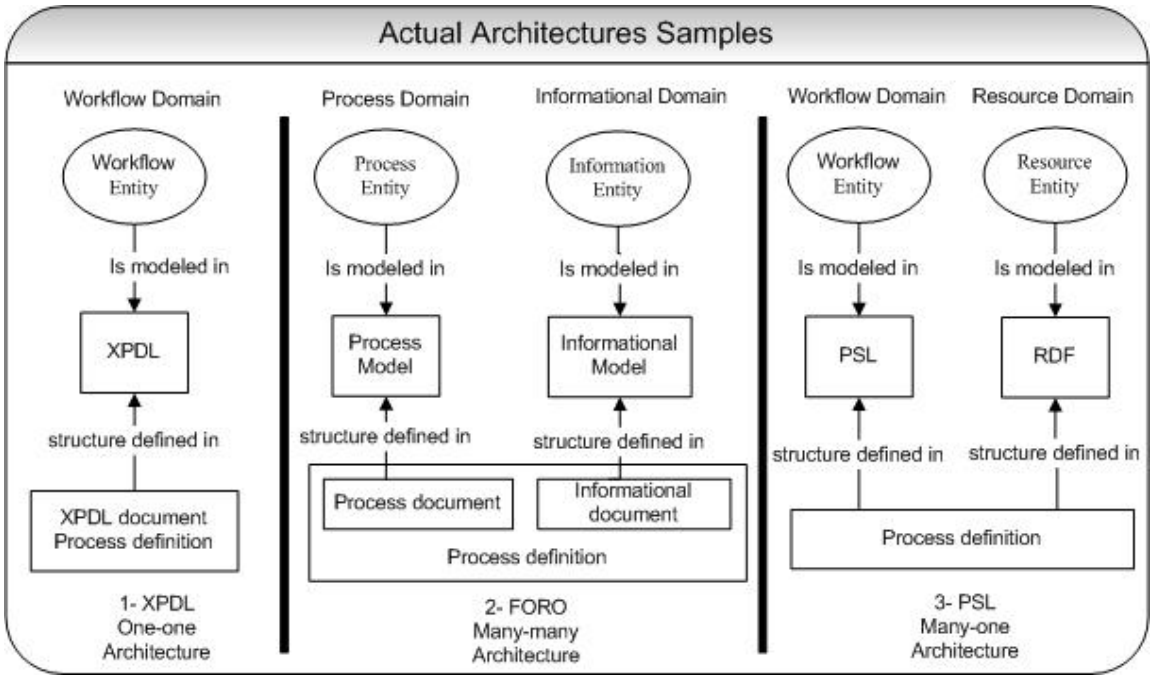

Fig. 2. Architecture's samples

4.1 XML Implementation

For implementing the process modeling architectures with XML technology it is necessary: (i) to define the models with a document type definition (DTD) or a XML Schema; and (ii) to implement applications for interpretation and manipulation of XML files.

In the *one-one* architecture, the process definition is a XML file that is in conformance with the only defined model.

In the *many-many* architecture, each document of a process definition is a XML file that is in conformance with the correspondent defined model. Another important implementation in this kind of architecture is the implementation of resources that make the document synchronizations. The document synchronizations are necessary for relationship maintenance that exists between elements of different documents.

In the *many-one* architecture, the definition of the integration rules of involved models is needed. An integration model can make the integration rules. To define the integration model in XML, a DTD or a XML Schema can be used. In the integration model, one of the integrated models is the main model and the others are the secondary models. The main model elements can be inserted normally in the integration DTD and the elements of the other models are inserted by namespace's standard (Generally, the integration model is an adaptation of main model). In the *many-one* architecture, the process definition is a XML file that is in conformance with the integration model. The figure 3 shows a more detailed *many-one* architecture.

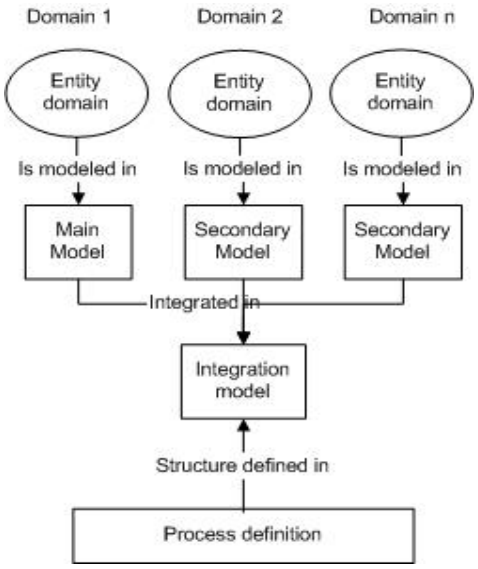

Fig. 3. Detailed *many-one* architecture

5 *Many-One* Case Study I - The Amaya Workflow Prototype

This *many-one* case study is presented to show more details about the *many-one* architecture application. The Amaya Workflow (AW)[16] [13] is a definition tool developed in Universidade Federal do Rio Grande do Sul (UFRGS). The AW was developed as an extension of Amaya [24] XML. The Amaya is a XML editor and browser, developed in Institut National de Recherche en Informatique et en Automatique (INRIA). The internal representation model of AW is similar to the model proposed by Casati et al. [5]. Following, some elements of AW model are described:

Workflow: It represents a process. It is the root element.
Task: It represents a workflow task.
Connector: It represents the connections between workflow elements.
MultiActivity and **SuperActivity**: They represent tasks that can be expanded.
The other elements are different types of joins and forks.

The internal representation of AW includes some SVG and Xlink attributes in the AW elements. In the beginning of internal representation, the namespaces of XLink, SVG and AW model need to be defined. The AW is the main model. XLink and SVG are secondary models. An example is shown following:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE AW PUBLIC http://www.inf.ufrgs/~telecken/AW/AW.dtd">
```

<workflow xmlns=" http://www.inf.ufrgs/~telecken/AW/"
xmlns:svg="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" version="1.0">

5.1 The Graphic Vision

Associated with each element of AW there is one graphic representation used in one AW graphic vision. The graphic representation is equivalent to the representation proposed in WIDE project [5]. The AW graphic vision is a functionality that helps the workflow designers in the workflow edition and workflow analysis.

The SVG attributes are used for storing of entities about the workflow graphic appearance. Each AW element contains SVG attributes that describe the correspondent graphic representation. For example, into the Task element of AW, the stroke, fill, x, y, width, Stroke-width and height attributes of element rect need to be inserted. The rect is an SVG element. To inform that these attributes belong to rect element, the type attribute with the value "rect" needs to be inserted. The type attribute is not a SVG attribute but it is used in this research. The table 2 shows what each inserted attribute represents. Each column contains information about the attribute informed in the table header. All attributes describe a rectangle.

Table 2 - The attributes of *rect*.

Stroke	stroke-width	fill	X	Y	width	height
Stroke color	Stroke width	Internal color	Coordinate X of left superior corner	Coordinate Y of left superior corner	Width	Height

An example of *Task* element is shown following.

<Task name="Fill document" ID="1234" application="WebForm" description="The manager fill the document"
svg:type=”rect” svg:stroke="black" svg:fill="white" svg:y="31px" svg:x="40px" svg:width="90px" svg:height="40px"
svg:stroke-width="2"/>

The SVG attributes are prefixed by "svg:". The other attributes are workflow attributes of AW model. The same procedure is applied on the other elements until all workflow graphic representation is defined by SVG attributes.

5.2 The Nesting

The AW has nesting functionality. The *SuperTask* and *MultiTask* elements aim to external process definitions. The process definition aimed by *SuperTask* or *MultiTask* describes the activities of these elements. The link of these documents is made by XLink attributes. Into the *MultiTask* or *SuperTask* elements of AW model, the type and href attributes of XLink simple element need to be inserted. To inform that these attributes belong to simple element, the type attribute with the value "simple" need to be filled. The value of href is a valid Uniform Resource Locator (URL) that aims to one process definition. An example of *SuperTask* element is shown following:

<SuperTask name="Fill document" ID="1234" application="WebForm" description="The manager fill the document"
svg:type=”rect” svg:stroke="black" svg:fill="white" svg:y="31px" svg:x="40px" svg:width="90px" svg:height="40px"
svg:stroke-width="10" xlink:type= “simple” xlink:href=”./exec.xml”/>

The two XLink attributes are prefixed by "xlink:".

5.3 Other Functionalities and the Namespace

Other functionalities of AW are: error verification; exportation to the XPDL, SVG and PDF formats; textual visions synchronized with graphic visions; internal representation controls; cooperation resources; etc. All AW functionalities use XML resources and the process definitions. The use of namespace standard does not disturb these functionalities. Basically, an attribute added by namespace is accessed in the same way as other attributes. The difference is the name of the attributes. The namespace attributes are prefixed by the model name and a colon sign (for example: xlink:href, svg:x). The elements of main model have only the attribute name. The figure 4 shows a screenshot of AW. The screenshot shows a graphic vision, a textual vision, a modeled workflow and a palette of workflow symbols.

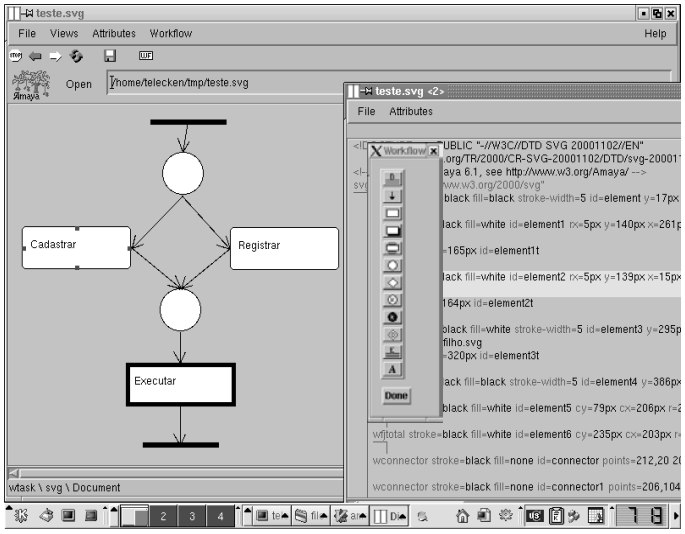

Fig. 4. The AW Screenshot

5.4 The *Many-One* Architecture of AW

The AW process definition can be divided into three distinct domains: workflow domain, graphic domain and link domain. These domains have many mutual relationships and, during the process definition edition, much synchronization is needed. The workflow, graphic and link domains are represented respectively by elements and attributes of AW, SVG and Xlink models.

An integration model was developed to integrate the models. In the integration model, the AW is the main model and the others are the secondary models. The secondary model attributes were inserted into main model elements by namespace standards. The AW process definition is formed by only one document. Into this document there are elements of AW model and attributes of AW, SVG and Xlink models. The figure 5 shows the *many-one* architecture of AW.

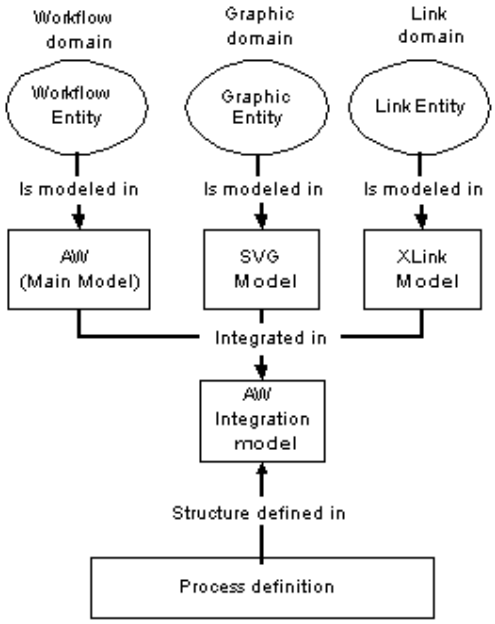

Fig. 5. The AW many-one architecture.

5.5 The Benefits of *Many-One* Architecture in the AW Project

The namespace divides more cleanly the elements of different domains. The elements that represent graphic entities are prefixed by "svg:". The elements that represent links are prefixed by "xlink:". The other elements are elements that represent workflow entities.

XLink and SVG are very diffused official W3C recommendations. In the AW project, documentation and several services available for these models (discussion lists, consultantship, etc) are used. These documentations and services facilitate the model learning, qualify the project and decrease the project costs.

Many applications and tools were developed for SVG and XLink. The AW is compatible and based in these applications and tools. Some features of Amaya system were used and extended in the AW development. The Amaya

system has the following components: a structured document editor that can edit XML document; a SVG viewer and editor; a namespace support and a XLink support. All these components were reused into AW system. Few adaptations were necessary. These reuses decrease the development cost and grow the compatibility of AW with XML, SVG and XLink applications.

The AW development is modular. It is possible to change any model (AW, SVG or XLink) and maintain the others. Also it is possible to add new XML models.

6 *Many-One* Case Study II - Adding Entities in XPDL Representations

In the case study I, an application of *many-one* architecture in one internal representation was described. In this case study, an application of *many-one* architecture in one interchange pattern is described. In this application, SVG attributes are added in XPDL documents.

Following, some XPDL elements are shown:

Package*:* It is a package that contains several processes.
ExternalPackage*:* It references to an external package.
WorkflowProcess*:* It represents a workflow process.
Activity*:* It represents a workflow process activity.
Transition: It represents a transition or a connector between other elements.

6.1 Instructions to Add SVG Attributes in XPDL Elements by the Namespace Standard

In the beginning of internal representation, the namespaces of SVG and AW model need to be defined. The AW is the main model and SVG is the secondary model. An example is shown following:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<Package xmlns="http://www.wfmc.org/2002/XPDL1.0"
xmlns:svg="http://www.w3.org/2000/svg"
version="1.0" Id="0" Name="Sample">
```

In the graphic representation of this case, icons represent the *Activity* elements. There are many types of Activity and for each type there is a correspondent icon. The *Transition* elements are represented by polylines.

Into the XPDL *Activity* elements, the following SVG attributes need to be inserted: *href*, *width*, *height*, *x* and *y* of element *image*. The value of *href* attribute is a URL of a file. The referred file contains an image that represents one type of activity in the graphic vision. The *width, height, x* and *y* elements define respectively the width, height and position of the icon in the graphic vision.

To inform that these attributes belong to *image* element, the *type* attribute with the value " image " needs to be filled. An example of *Activity* element is shown following:

```
<Activity Id="5" Name="Email Confirmation" svg:type=”image” svg:width =”90” svg:height = “40”
svg:x="100" svg:y="100" svg:href=”../activity.jpg”>
    <Implementation>
                    <No/>
    </Implementation>
</Activity>
```

Into the XPDL *Transition* elements, the following SVG attributes need to be inserted: *stroke*, *stroke-width* and *points* of element *polyline*.

To inform that these attributes belong to *polyline* element, the *type* attribute with the value " polyline " needs to be filled. An example is shown following:

```
<Transition Id="22" From="1" To="12" svg:type="polyline" svg:stroke="black" svg:points="326,158 320,234
328,221 320,234 313,220" svg:stroke-width="2" >
   <Condition>status == "Valid Data"</Condition>
</Transition>
```

6.2 Adding other entities

An XPDL model can be divided into two distinct domains: the workflow domain and the simulation domain. The elements of simulation domain describe data used by workflow simulation software. The elements of workflow domain describe a workflow and are used mainly by workflow engines.

The simulation elements could be removed from XPDL model. A new model that defines only simulation elements could be created. And the new model elements could be inserted into XPDL documents by namespace standard.

In several XPDL points there are references from URLs. These references are links. The links could be removed from XPDL model and XLink attributes could be inserted into XPDL documents by namespace standard.

The PSL [14] proposes the use of RDF attributes to describe resources used in activities. These attributes could be inserted into XPDL documents by namespace.

An example of Activity code that contain elements and attributes of referred models (XPDL, SVG, XLink, RDF and simulation model) is shown following. The attributes and elements are inserted by namespace standard.

```
<Activity Id="5" Name="Email Confirmation"
     svg:type=”image” svg:width =”90”    svg:height = “40” svg:x="100" svg:y="100"
     xlink:type="simple" xlink:href=”../activity.jpg”
     rdf:resource="email.rdf#confirmation">
   <simulation:SimulationTransformation Instantiation="ONCE">
            <Cost>12<Cost/>
   <simulation:SimulationTransformation/>
   <Implementation>
            <No/>
   </Implementation>
</Activity>
```

The figure 6 shows the architecture of this case.

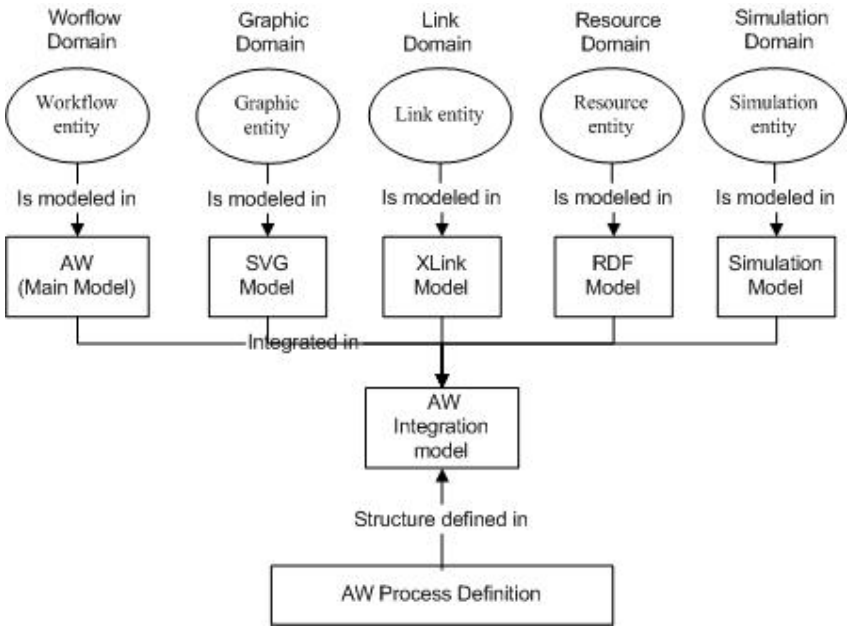

Fig. 6. Case II architecture

The objective of this example was to show the modularity, flexibility and possibility of *many-one* architectures. It is possible to make many other model combinations. Defining the best or more necessary model combination is not approached in this paper but it is an important future work.

7 Comparing the Architectures

The main advantage of the *one-one* architecture is that it is simpler to use because it implements and develops only one model and only one document. However, the current process definitions are very complex documents. There are great demands for inclusion of new elements in the process definition models. For each element that is inserted in the process definition, the model complexity grows. The complexity is propagated for all workflow components directly or indirectly involved with the process definition models (development, implementation, learning and use of workflow technology). And the cost of workflow technology grows too.

In some cases, separating the elements into many models can decrease the workflow technology costs. This is recommended mainly when the elements of model can be divided into distinct domains. The separation is possible in the *many-many* and in the *many-one* architecture. The main advantages of these architectures are:

- The developing of models in an independent way. With an independent development it is possible to divide a big problem into several smaller ones. The big problem is the developing of one model that has elements of several domains. The smaller problems are many models that can be developed in a more independent way and that can be separated into different domains. This independent development also includes all that is developed around the models (application, tools, patterns, learning, training, etc).
- Existent models can be used and reused in a more modular way. The structure of process definition can be formed by elements of different domains, and for each domain the developers can choose to use an existent model or to create a new one. If there are problems with one model, only this model needs to be changed.

The advantages of *many-many* and *many-one* architecture are similar. But the disadvantages are different. The disadvantage of *many-many* architecture is the need to develop resources for document synchronization. The disadvantage of *many-one* architecture is the need to develop an integration model. The table 4 shows the disadvantages and advantages of related architectures.

Table 4 – Architecture's advantages and disadvantages

Architecture	Advantages	Disadvantages
One-One	To use one document and one model (it is the simplest architecture).	To develop in one monolithic way.
Many-Many	To develop models and all technology involved with the models in one independent way. To reuse existent models and technologies in a more modular way.	To develop resources for document synchronization.
Many-One	The same as *many-many*	To develop an integration model.

For choosing architecture, it is important to know two characteristics of the process definition: the domain areas involved and the necessary synchronization among elements from different domains.

The domain areas and the distinction of domain areas are the characteristics that define if it is more appropriate to use an architecture with one model or with many models.

The necessary synchronization among elements of different domains is the characteristic that defines if it is more appropriate to use an architecture with one document or with many documents.

The following recommendations were defined by a comparison among these process definition characteristics and the presented process modeling architecture characteristics:

- If the **distinction of domain areas involved in one process definition decreases**, the use of *one-one* architecture is more appropriated. It is coherent to maintain entities of the same domain in a same model.
- If the **distinction of domain areas involved in one process definition grows**, the use of *many-many* or *many-one* architecture is more appropriated. It is coherent to maintain entities of different domains in different models.
- If the **need of synchronization among elements of different domains grows**, the use of *one-one* or *many-one* architecture is more appropriated. The costs of synchronization are greater when there are elements among different documents.
- If the **need of synchronization among elements of different domains decreases**, the use of *many-many* architecture is more appropriated. The costs of synchronization are greater when there are elements among different documents.

The table 5 shows the application of these recommendations. In the first column the architectures are shown. In the second column the characteristics of process definition appropriated for the associated architecture are shown.

Table 5 – Architecture's recommendations

Architecture	Characteristics of process definition appropriated
One-One	The entity domains are from the same domain
Many-Many	The entity domains are from different domains **and few synchronizations** are necessary
Many-One	The entity domains are from different domains **and many synchronizations** are necessary

8 Applying the Recommendations on Complex Process Definition Models

Many current complex process definitions contain the conditions for *many-one* architecture recommendation. These process definitions contain entity domains from different domains and many synchronizations are necessary in the process definition edition. This is the situation of the case studies shown in this paper. The process definition of the first case study can contain elements or attributes of workflow, graphic and link domain. The process definition of the second case study can contain elements or attributes of workflow, graphic, link, resources and simulation domain. In both case studies, many synchronizations are needed during the process definition edition.

For such situations, the *many-many* is the most onerous architecture. In the process definition edition, the synchronization and maintenance costs are very great for so much relationship among different documents. For using *many-many* architecture it is needed to have no synchronization or few synchronization.

Using the *one-one* architecture provides a more monolithic development. This type of development provides a solution more exact and specific for each application. However the reuse is low. For each application one exact, specific and monolithic solution is needed. It is more difficult to reuse just a part of model or just a part of solution.

If the complexity and the quantity of entities grow very much, a monolithic solution can be very onerous or unviable. In this case, modular solutions such as *many-one* architecture can be more efficient. Modular solutions can divide a great problem into several little ones.

The *many-one* architecture provides a more modular and reusable workflow technology development. For example, in the second case study shown in this paper, groups of developers and researches could work exclusively in the XPDL model. Other groups could work exclusively in each one of other technologies (SVG , XLink, RDF and simulation).

The technology developed by groups dedicated to the SVG, Xlink, RDF and simultation can be reused by: (i) other interchange patterns, such as BPML and BPEL4WS;(ii) other internal representations, such as Biztalk Orchestration Designer internal representation; and (iii) any other application inside or outside of workflow area, such as SVG viewers and Extensible Hypertext Markup Language (XHTML) links.

Integration groups also are important. These groups make the integration of different modules and technologies. In the first case study, a group that make the integration of AW, SVG and XLink is needed. In other projects, other groups can integrate BPEL4WS with XLink, PSL with RDF (this is the case of PSL [14]), XPDL with SVG , XLink, RDF and simulation (this is the situation of second case study), etc.

9 Conclusion

Following, some *many-one* recommendations are presented. These recommendations are a summarization of the main contributions of this paper:

1. It is recommendable to use the *many-one* architecture in a complex process definition application that contains entity domains from different domains and when much synchronization among elements from different domains is necessary. Many current process definitions have these characteristics. Many complex environments need process definition with such characteristics.
2. When a more modular and independent development is needed, it is recommendable to use or to consider the *many-one* architecture.
3. For optimizing the *many-one* architecture benefits it is recommendable to reuse current XML models and reuse all technology developed around these models (application, tools, APIs, services, documentation, resources, researches, involved community, implementation, technologies, etc).
4. As the use of *many-one* architecture (mainly using these recommendations) grows, the *many-one* architecture benefits grow too. In an ideal scenario there are many models for different domains. There are many integration developers and researches that can group and regroup the available models (and the involved technology) in according with the specific application needs.

The ideal proposed scenario is not a distant scenario. In the current days, there are many XML models available. The model integration technology is available too. Some applications using namespace architecture already were implemented.

But it is necessary to organize and to optimize this scenario. This can be made by more researches about model integration technologies, the use of XML models in process modeling phase and process modeling architectures.

Finally, it is expected that this paper: (i) have presented the main ideas, fundaments and recommendations about *many-one* architecture; and (ii) make developers, researches and organizations, such as WfMC and BPMI, aware about the importance of optional process modeling architectures. In complex environments, the three solutions (*one-one, many-one* and *many-many*) need to be considered.

Acknowledgements

The authors want to tank the support of CNPq, UFRGS, INRIA and Banco Central do Brasil.

References

[1] van der Aalst, W.M.P. Don't go with the flow: Web services composition standards exposed, *IEEE Intelligent Systems*, 18(1):72-76, 2003.

[2] BPMI, Business Process Management Initiative Home Page. http://www.bpmi.org

[3] Bray, T.; Hollander, D. and Layman, A. Namespace in XML, W3C Recommendations, 1999. http://www.w3.org/TR/REC-xml-names.

[4] Bray, T.; Paoli, J.; Sperberg-McQueem, C.M. and Maler, E. eXtensible Markup Language (XML) 1.0 (Second Edition), 2000. http://www.w3.org/TR/REC-xml.

[5] Casati, F.; Grefen, P.; Pernici, B.; Pozzi, G. and Sanchez, G. WIDE Workflow Model and Architecture, Technical Report 96-19, Centre for Telematics and Information Technology (CTIT), University of Twente, Netherlands, 1996.

[6] DeRose, S.; Maler, E. and Orchard, D. XML Linking Language (XLink) Version 1.0. W3C Recommendation, 2001. http://www.w3.org/TR/xlink/

[7] Dodds, D.; Watt, A.; Birbeck, M.; Cousins, J.; Moore, D.R.; Worden, R. ; Nic, M.; Ayers, D.; Ahmed, K.; Wrightson, A. and Lubell, J. Professional XML Meta Data, Wrox press, Birmingham, AL, 2001.

[8] FORO, Models Designer Manual, 2001. http://www.foro-wf.com/docs/english/ModelDesigner2.1.3.pdf

[9] IBM. IBM MQ series Workflow Programming Guide Version 3.3, IBM Corporation, Armonk, USA, 2001.

[10] ILOG Inc, Ilog components for business process management solutions, 2001. http://www.ilog.com/products/jviews/workflow/workflow_wp.pdf

[11] Leymann, F. Web Services Flow Language (WSFL), Technical report, IBM, 2001.

[12] Leymann, F. and Roller, D. A quick overview of BPEL4WS, IBM DeveloperWorks, August 2002.

[13] Pinheiro, M.K.; Telecken, T.L.; Lima, J.V.; Zeve, C.M.D. and Edelweis, N. A Cooperative Environment for E-Learning Authoring. Document Numérique, França, v.5, n. 3-4, p. 89-114, 2002.

[14] Schlenoff, C.; Gruninger, M.; Tissot, F.; Valois, J.; Lubell, J. and Lee, J. The Process Specification Language (PSL): Overview and version 1.0 specification, NISTIR 6459, National Institute of Standards and Technology, Gaithersburg, MD, 2000.

[15] Sheth, A.P.; Georgakopoulos, D.; Joosten, S.; Rusinkiewicz, M.; Scacchi, W.; Wileden, J.C. and Wolf, A.L. Report from the NSF Workshop on Workflow and Process Automation in Information Systems, SIGMOD Record 25(4) (1996) 55-67.

[16] Telecken, T.L.; Lima, J.V.; Zeve, C.M.D.; Maciel, C. and Borges, T. Modeling of Courses through Workflow using the standard SVG/XML. In Proceedings of EDMEDIA'2002 World Conference on Educational Multimidia, Hipermidia & Telecomunications, Denver, USA, 2000, 24-29

[17] Thatte, S. XLANG. Web Services for Business Process Design, Technical report, Microsoft Corporation, 2001.

[18] Workflow Management Coalition Home Page. http://www.wfmc.org

[19] Workflow Management Coalition. Interface 1 - Process Definition Interchange. Technical report WFMC-TC-1025, 2002.

[20] Workflow Management Coalition. Terminology and Glossary, Technical report, WFMCTC-1011, Brussels, 1996.

[21] World Wide Web Consortium. Resource Description Framework (RDF), W3C Recommendation, 1999. http://www.w3.org/TR/REC-rdf-syntax

[22] World Wide Web Consortium. Scalable Vector Graphics (SVG) 1.0 Specification. W3C recommendation, 2001. http://www.w3.org/TR/SVG/

[23] World Wide Web Consortium. Web Service Choreography Interface 1.0, 2002. http://www.w3.org/TR/wsci/

[24] World Wide Web Consortium. Welcome to Amaya. http://www.w3.org/Amaya/