DESIGN AND IMPLEMENTATION OF A BUSINESS PROCESS REPRESENTATION MODULE[*]

Costin BADICA
Department of Computer Science
King's College, Strand, London, WC2R 2LS
badica@dcs.kcl.ac.uk

Chris FOX
Department of Computer Science
University of Essex, Colchester, CO4 3SQ, UK
foxcj@essex.ac.uk

Abstract. *This paper reports on the work done in the INSPIRE project on developing the Process Representation Module (PRM hereafter). The major aim of INSPIRE is the development of a tool to support a more intelligent an human-oriented approach to business process re-engineering (BPR hereafter). Our task was to develop the PRM, which is a core module of the INSPIRE tool. The main responsibility of the PRM is to provide an all-encompassing and consistent representation of business processes ([12]). The paper contains roughly the following information: a description of the architecture and the general idea of the INSPIRE tool; some definitions of key terms related to business processes, as found in the literature; a brief description of some well-known formalisms for business process modelling and of the formalism employed in INSPIRE; architecture and data models for the PRM; details of the PRM design and implementation.*

Keywords: *business process, software architecture, data model, Prolog programming language*

Introduction

The aim of the INSPIRE project is to develop a tool to support a more intelligent and human-oriented approach to BPR. King's College London is part of the project consortium. It is responsible for developing 2 modules: PRM and a Natural Language Interface (NLP hereafter).

The INSPIRE partners are:

Two major manufacturing end users:
- Swan Hunter Ltd, a shipyard
- Drahtcord Saar, a steel cord supplier

Two management consultancies who are BPR experts:
- Central Consulting Group (CCG)
- EFESO consulting

Three software providers:
- British Maritime Technology
- TXT E-Solutions
- King's College London

Definition of Key Terms

An increased interest in applying information technology to the field of business processes has been manifested during the last decade, both in research and commercial communities, as a response to the BPR slogan. This statement is proven by the large number of papers published on the subject, the large number of emerging standards and standardization proposals for representing different aspects of business processes, and the large number of software tools that support tasks like business process modelling, design, analysis and simulation.

[*] The work described here was supported by funding from the European Union as part of the Framework V INSPIRE project (IST-10387-1999)

Informally, by a business process (BP hereafter) we mean a process that is carried out in an organization in order to achieve the organization's business objectives. An organization usually runs more than one business process. Some examples of business processes are: handling order for goods, recruiting staff, designing a new product, building a software system according to a specification, installing a new telephone service, a.o. Ould ([1]) identifies three major types of BPs: core processes, support processes and management processes.

There are also a lot of definitions in the literature on the subject of business processes:

i) A BP is a set of partially ordered activities (carried out inside an organization) intended to reach a (business) goal ([3]);

ii) A BP is "a set of related activities that produces a result of value for the customer" and a "specific ordering of work activities across time and place with a beginning and clearly defined inputs and outputs" ([2]);

iii) Ould ([1]) avoids to give a precise definition of what a BP is. Rather, he describes the key features of a thing that he calls a BP: it contains a set of activities done for some reason; the activities are carried out by people and/or machines it is carried out collaboratively by a group; it often crosses functional boundaries; a functional unit is here understood as a part of an organization with specific responsibilities like personnel, manufacturing, marketing, finance, a.o. it is driven by the outside world (reactive in software engineering terms ?); i.e a BP has generic customers or clients

We can conclude that a BP is set of logically interrelated activities in an organization carried out by people and/or machine in order to contribute to the achievement of the business objectives of the organization. So, a BP has: *activities (*also called *tasks), participants* and *goals*. The participants are artefacts (products, machines) and/or human beings (customers, workers). We must distinguish between *active participants* that execute activities and *passive participants*, i.e. the ones the process activities are directed to. The active participants are sometimes called *resources* and are organized in *resource pools*.

Business process modelling (BPM hereafter) is the task of producing an abstract description of a business process. The BP can be an existing process running in an organization or a new process existing only in the modeller mind.

Curtis, Kellner and Over ([5]) mention the following purposes of BPM:

- to facilitate human understanding and communication;
- to support process improvement, which is based on a formal or informal assessment of the process model;
- to support process management, i.e the process model can be the basis for planning, monitoring and coordination of the underlying BP;
- to automate process guidance by capturing and reusing the process know-how;
- to automate process execution support, for example with the help of a workflow management system. According to Wieringa ([4]), an workflow management system monitors the tasks to be performed in an organization.

A BPM is expressed by means of a representation formalism. One such formalism should be able to capture all the related information from the model and additionally should facilitate the analysis of the modelled process (for example, in a BPR context).

The classical definition of BPR given by Hammer in [3] states that BPR is the radical redesign of business processes, usually enabled by information technology, in order to achieve dramatic improvements in their performance. Some examples of performance indicators of a BP are: cost, quality (of the products and/or services provided), speed, productivity, a.o.

The BP field is of interest for the software engineering community for several reasons. One reason is that the software process can be thought of as a BP with the goal of developing a program according to a set of specifications. Maybe this is why most of the BPM formalisms have their origin in the research on modelling the software process.

The Inspire Tool

BPR is intended to manage the change. According to INSPIRE, BPR is a structured approach towards change of the current situation – *As Is,* to a new situation – *To Be,* with the goal to achieve some benefits from this.

A BPR process is usually broken into a sequence of stages, including: understanding and capturing the current situation as an *As Is*, search for a better solution, i.e one or more improved *To Be* processes, and implementation, i.e. produce an implementation plan and then run it.

The INSPIRE toolset has a modular architecture. It is represented in simplified form in figure 1. There is also a repository implemented as a server. The repository is meant to store successful old BPR projects (it is not shown in figure 1). So we have a 4-tiers architecture comprising: the upper-level modules, including various graphical editors and simulators; the intermediary interface including wrappers; the wrappers provide suitable views for the upper level modules; the PRM as a central data model; the repository to store best practices.

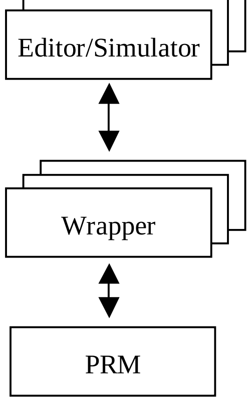

Figure 1.Architecture of INSPIRE

The Inspire Formalisms For Modelling BPs

We have looked at some existing formalisms before coming up with the formalism employed within INSPIRE. The criteria for selecting them were: their availability as standards, their use into existing tools, their description in reference papers/books on the subject and ultimately the experience with using them by the project partners (consultants and developers, mainly). Some of these formalisms are:

- the IDEF0 and IDEF3 modelling languages from the IDEF suit;
- the Role Activity Diagramming (RAD) notation employed by the STRIM method, described in Ould’s book (1995); at some point there was a proposal from one of the project partners to use a variant of the RAD notation, called AAD (Actor Activity Diagramming), as another view of a business process.

IDEF0 is based on SADT developed by Douglas Ross. The method was initially used (together with the other techniques IDEF1 and IDEF3) for modelling manufacturing processes. Then, during the 90s, the IDEF suit was developed by NIST into a set of FIPS standards.

IDEF0 is used to produce a *function model* of the system. This is a structured representation of the functions, activities or processes within the modelled system or subject area. IDEF0 assists the modeller in identifying what functions are performed and what is needed to perform those functions. The characteristics of IDEF0 are:

- a model is a set of diagrams;
- a diagram is composed of labelled boxes and arrows;
- a box models an activity or task;
- an arrow models a flow of data or objects from source to use (destination);
- there are four kinds of arrows: inputs and outputs (that show what is consumed and produced by an activity), controls (that show when an activity can be executed)

and mechanisms (that show what is needed for the activity to be executed).
- it provides a gradual exposition of detail, through hierarchical decomposition.

IDEF3 is used to produce a *dynamic model* of the system. Its characteristics and elements are:
- a model is a set of process schematics;
- a process schematics is composed of units of behaviours (UOB hereafter), links and junctions;
- an UOB models an activity or event;
- a link models a causality or precedence relation between UOBs;
- the use of junctions specify the logic of process branching.

The formalism employed within INSPIRE is a hybrid between IDEF0 and IDEF3, complemented with additional features for modelling human resources information. It combines the functional and hierarchical decomposition aspects of IDEF0 with the dynamic ones of IDEF3. The idea is to come out with both a black box and a glass box view of an activity. The black box view is based on IDEF0. The glass box view is based on IDEF3 and it acts as a refinement of the black box IDEF0 view. In order to achieve this, each activity is complemented with a tree of input junctions (connectors, in the INSPIRE terminology) and with a tree of output junctions. Also the formalism is complemented with facilities for representing human resource information, including information about agents, skills, competencies and profiles.

Let us consider the example of a manufacturing company. We can recognize the existence of a BP for material acquisition. This process takes material requests and produces purchase orders and payment authorizations. We can identify a sub-process for processing the material requests. The company has a list of available suppliers, but obviously is must always be prepared to find and handle new potential suppliers.

The Inspire Formalisms For Modelling BPs

We have looked at some existing formalisms before coming up with the formalism employed within INSPIRE. The criteria for selecting them were: their availability as standards, their use into existing tools, their description in reference papers/books on the subject and ultimately the experience with using them by the project partners (consultants and developers, mainly). Some of these formalisms are:
- the IDEF0 and IDEF3 modelling languages from the IDEF suit ([10], [11]);
- the Role Activity Diagramming (RAD) notation employed by the STRIM method, described in Ould's book ([1]); at some point there was a proposal from one of the project partners to use a variant of the RAD notation, called AAD (Actor Activity Diagramming), as another view of a business process.

IDEF0 is based on SADT developed by Douglas Ross. The method was initially used (together with the other techniques IDEF1 and IDEF3) for modelling manufacturing processes. Then, during the 90s, the IDEF suit was developed by NIST into a set of FIPS standards. IDEF0 is used to produce a functional model of the system, while IDEF3 is used to model the system's dynamics as sequences of activities.

IDEF0 ([10]) is used to produce a *function model* of the system. This is a structured representation of the functions, activities or processes within the modelled system or subject area. IDEF0 assists the modeller in identifying what functions are performed and what is needed to perform those functions. The characteristics of IDEF0 are:
- a model is a set of diagrams;
- a diagram is composed of labelled boxes and arrows;
- a box models an activity or task;
- an arrow models a flow of data or objects from source to use (destination);

- there are four kinds of arrows: inputs and outputs (that show what is consumed and produced by an activity), controls (that show when an activity can be executed) and mechanisms (that show what is needed for the activity to be executed).
- it provides a gradual exposition of detail, through hierarchical decomposition.

IDEF3 ([11]) is used to produce a *dynamic model* of the system. Its main characteristics are:

- a model is a set of process schematics;
- a process schematics is composed of units of behaviours (UOB hereafter), links and junctions;
- an UOB models an activity or event;
- a link models a causality or precedence relation between UOBs; junctions specify the logic of process branching.

The formalism employed within INSPIRE is a hybrid between IDEF0 and IDEF3, complemented with additional features for modelling human resources information. It combines the functional and hierarchical decomposition aspects of IDEF0 with the dynamic ones of IDEF3. The idea is to come out with both a black box and a glass box view of an activity. The black box view is based on IDEF0. The glass box view is based on IDEF3 and it acts as a refinement of the black box IDEF0 view. In order to achieve this, each activity is complemented with a tree of input junctions (connectors, in the INSPIRE terminology) and with a tree of output junctions. Also the formalism is complemented with facilities for representing human resource information, including information about agents, skills, competencies and profiles.

Let us consider the example of a manufacturing company. We can recognize the existence of a BP for material acquisition. This process takes material requests and produces purchase orders and payment authorizations. As part of it we can identify a sub-process for processing the material requests. The company has a list of available suppliers, but obviously is must always be prepared to find and handle new potential suppliers. This example is shown in figure 2.

We are using radar charts (also known as spider-web charts) to model the competencies / skills / profiles of human agents. Radar charts are useful when you want to look at several different factors, all related to one item. A radar chart has multiple axes along which data can be plotted.

In INSPIRE radar charts are combined with a tree representation of profiles, thus obtaining a sort of nested radar charts. A profile of a person is represented as a tree of competences. For example, we can evaluate a Software Engineer according to the following competences: formal methods, communication, design and programming skills. Then, we can refine programming into C++, Java and Prolog and design into structured design and object-oriented design. Thus we obtain a complex structure for a profile.

To evaluate a person according to a given profile we must provide a score for the leaf competencies of the profile tree. The scores of the rest of the competencies are computed using averaging formulas from the scores of the children competencies. The score information is used in the design of the *To Be* process, to compute the gap between the evaluation of the profile required by an activity and the evaluation of a given person profile.

The PRM Data Model And Architecture

The PRM stores the following pieces of information:

- *process information*, i.e. activities, flows, and decompositions;
- *human resource information*, i.e. agents, competencies, profiles and assignments of agents to activities;
- *simulation information*, i.e. the values of the performance indicators, as computed using simulation (work in progress);

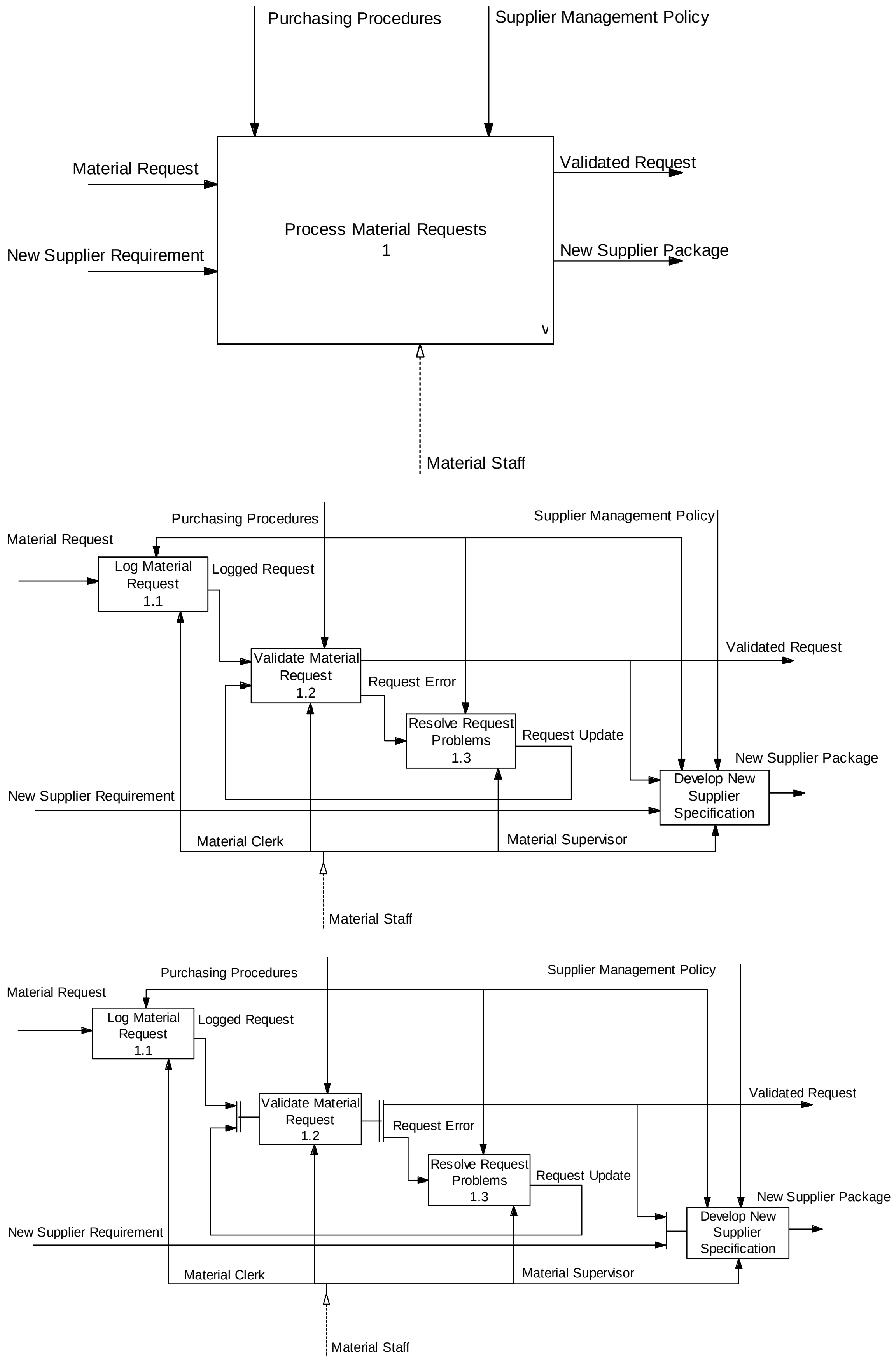

Figure 2.An example of a business process modelled with the INSPIRE tool

Due to the lack of space, we reproduce here only the data models of the process information, comprising and IDEF0 part and an IDEF3 part.

The data model for the IDEF3 and IDEF0 process information can be represented as the class diagrams shown in figures 3 and 5.

When devising the PRM architecture we taken into account the following facts:

- the PRM must provide the C++ API expected by the rest of the modules of the INSPIRE application (graphical editors, simulators and wrappers). The complexity of this API is quite high. It contains more than 40 classes, each with at least of 5 interface functions (many with more than 10 !).
- the PRM must be capable to acquire and produce a Prolog representation of a business process, as required by the NLP module.

Thus, we obtained the layered architecture shown in figure 4.

The PRM Implementation Details

The PRM internal representation is based on a very simple object model. The model was inspired by the Resource Description Framework (RDF hereafter) data model ([6],[7]). RDF is a model proposed by the World

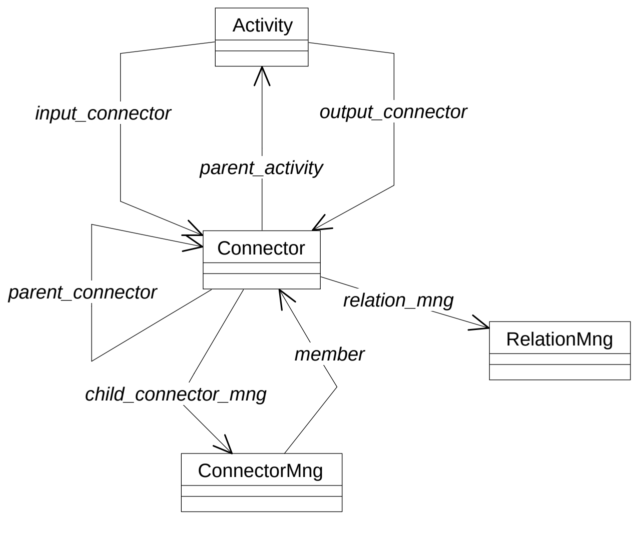

Figure 3.Data model for IDEF3

Wide Web Consortium for representing metadata about the resources of the web.

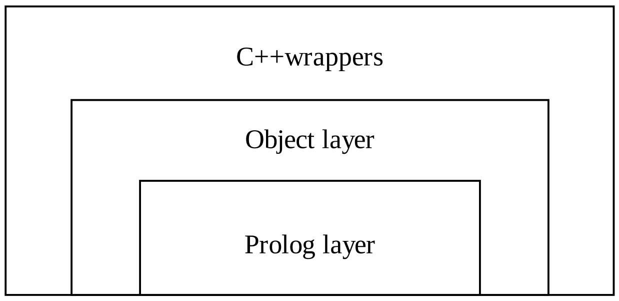

Figure 4.Layered architecture of the PRM

According to RDF, the world is modelled as a set of resources. So, everything is a *resource*. A resource has named properties and properties have values. A value can be either a literal or another resource. Each resource has a unique identity represented by an Uniform Resource Identifier, or URI for short. This model is very flexible and can practically support any process representation modelled as a class diagram.

We are using the term *object* instead of *resource*. A model is a set of triples ⟨*object id*, *attribute name*, *attribute value*⟩. A triple states that the object indicated by the first argument has the value given by the third argument for the attribute with the name given as the second attribute.

So, a model is just a collection of triples. Each triple is represented as a Prolog fact with the following relation scheme:

oav(*ObjectID*,*AttributeName*,*AttributeValue*)

The *AttributeName* is a unary functor of one of the forms:

l(*StringValue*), if the value is primitive (i.e. represented as a string)

o(*ObjectId*), if the value is another object

For example, the representation of an activity in our model is follows (note that some attributes are omitted here):

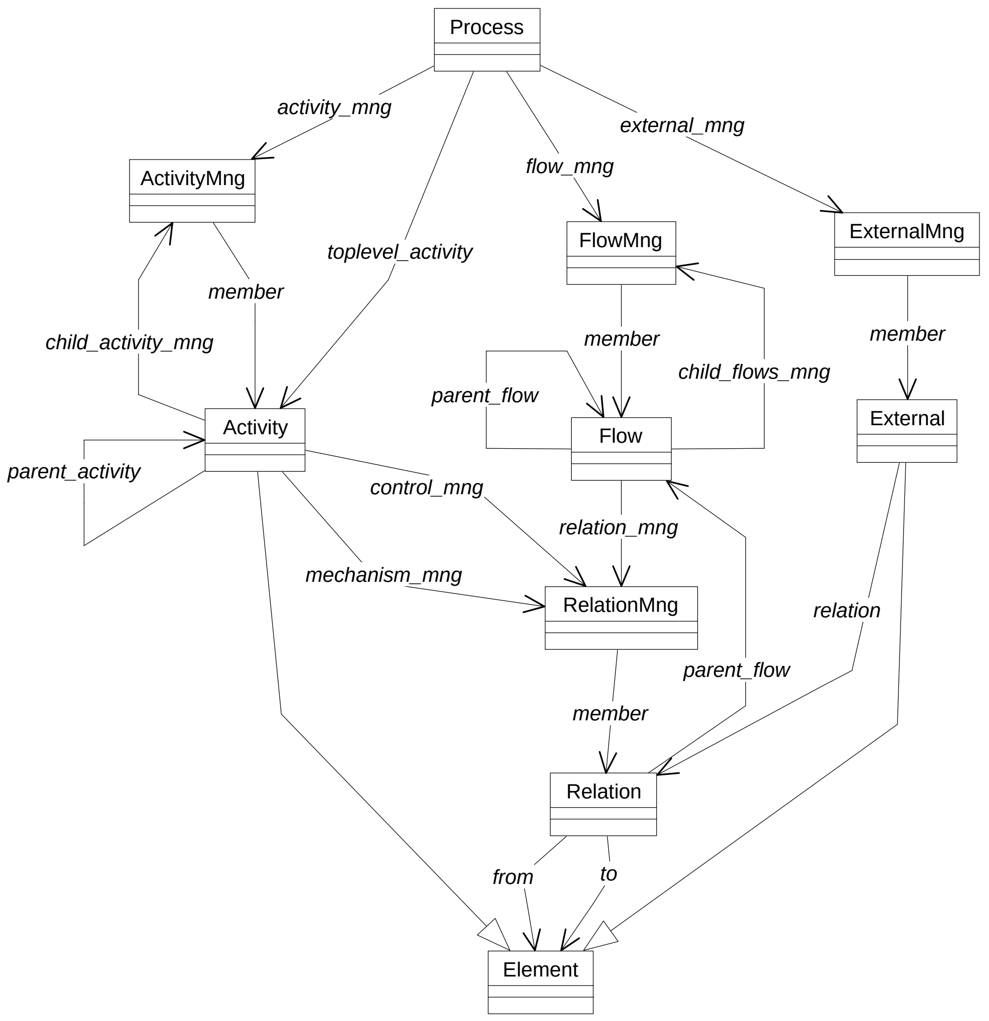

Figure 5.Data model for IDEF0

```
oav(46,y_coordinate,l('201')).
oav(46,x_coordinate,l('405')).
oav(46,cost,l('0')).
oav(46,max_iter,l('1')).
oav(46,number,l('1')).
oav(46,duration,l('1.000000')).
oav(46,notes,l('Input notes')).
oav(46,description,l('Input
description')).
oav(46,name,l('Root')).
oav(46,id,l('7')).
oav(46,mechanism_manager,o(64)).
oav(46,control_manager,o(63)).
oav(46,child_activity_manager,o(62)).
oav(46,output_connector,o(56)).
oav(46,input_connector,o(50)).
oav(46,type,l('Activity')).
```

For each object stored as triples there is a unique C++ wrapper object in the upper layer. This wrapper object stores only the object ID, to be able to identify the object from the Prolog side. Additionally, for each object there is a triple that gives the address of the corresponding C++ wrapper object.

The object layer implements the base class for the C++ wrappers and also the functionality required by the manager objects, which are a special kind of objects. The C++ wrapper objects provide the interface expected by the other modules of the INSPIRE application.

A BP model can become very big. For example, for a process with 100 activities and without any flows and human resources information the PRM creates more than 2000 of objects, and more than 11000 of Prolog facts !

The PRM was implemented as a DLL on a Windows NT platform using Visual C++.

As Prolog engines, we considered Amzi Prolog and SWI-Prolog. They are available as DLLs.

We started with Amzi Prolog in the first stage. Amzi Prolog provides a very simple C++ interface. It can be easily embedded as a *logic server*. The logic server is provided as a class. So, you can embed as many logic engines as you want in your C++ code, by simply instantiating this class. The logic server class provides a set of services, for querying the logic engine and for retrieving the answers of the query. However, as soon as we were able to make tests with creating and updating reasonable large process models, we have noticed that Amzi Prolog 5 has serious problems with efficiency. The cost of an update significantly increased with the size of the model. So, when the model became quite large the behaviour of the graphical editors became annoying and we concluded that Amzi Prolog 5 was unusable to do the updates on the logic base synchronously with the user input. One source of the problem was the lack of indexing provided by Amzi Prolog 5. However, this problem has been resolved in Amzi Prolog 6.

We have also considered SWI-Prolog ([8]). SWI-Prolog comes with a C-interface. Recently a C++-interface was provided. The first experiments with SWI-Prolog and its C++ interface are encouraging. The API is good, and the support for casting between Prolog types and C++ types is better than of Amzi Prolog. Also the speed is incomparable with that of Amzi Prolog 5 and even better then of Amzi Prolog 6. We hope to handle with SWI-Prolog/Amzi Prolog 6 models consisting of up to 100000 of Prolog facts.

References

[1] Ould, M. (1995) *Business Processes: Modelling and Analysis for Reengineering and Improvement*, John Wiley & Sons.

[2] Davenport, T.H. (1993) *Process Innovation: Reengineering Work Through Information Technology*, Harvard Business School Press

[3] Hammer, M., Champy, J. (1993) *Reengineering The Corporation: A Manifesto For Business Revolution*, HarperBusiness

[4] Wieringa, R.J. (2000) *Design Methods for Reactive Systems: Yourdon, Statemate and the UML*, Department of Computer Science, University of Twente

[5] Curtis, B., Kellner, M.I., Over, J. (1992) *Process Modeling*. In: Communications of the ACM 35 (9) 75-90

[6] *RDF Model and Syntax*, W3C Recommendation, http://www.w3.org/TR/REC-rdf-syntax/

[7] *RDF Schema Specification 1.0*, W3C Candidate Recommendation, 2000, http://www.w3.org/TR/2000/CR-rdf-schema-20000327/

[8] SWI-Prolog Homepage, http://www.swi.psy.uva.nl/projects/SWI-Prolog/

[9] Amzi Prolog Homepage, http://www.amzi.com

[10] Draft Federal Information Processing Standards Publication 183 (1993) *Integration Definition for Function Modelling* (IDEF0)

[11] Mayer, R. et al.(1995) *Information Integration for Concurrent Engineering (IICE): IDEF3 Process Description Capture Method Report*

[12] Fox, C. (2000) *The Process Representation Module (Specification)*, INSPIRE (IST-1999-10387) Deliverable 2.1