Towards Cross Language Process Model Reuse – A Language Independent Representation of Process Models

Khurram Shahzad, Mturi Elias, and Paul Johannesson

Department of Computer and Systems Science (DSV),
Royal Institute of Technology (KTH) / Stockholm University, Sweden
{mks,mturi,pajo}@dsv.su.se

Abstract. Process model reuse is becoming a key approach to addressing the challenges of modeling business processes from scratch. A repository is, therefore, essential to store and manage process models for future reuse. In this paper, we develop a logical data model that enables a Universal Process Repository to store process models in the form of process elements, independent of any process modelling language. In order to store process models in the process repository we propose an algorithm that automatically extracts data from the repository and converts them to process models on the fly. Finally, we use a case study to present data stored about a process model in the repository and to illustrate the development of process models from the data stored in the repository.

1 Introduction

The reuse of process models can help business users simplify the work of modeling business processes [1], improve efficiency as well as substantially reduce the cost of modeling business processes [2, 3]. A repository is, therefore, necessary to store and manage process models for future reuse. The available process repositories (like IBM process repository [4], IBM BPEL repository [5] and SAP Business Maps [6]) are not publically open for change and growth, which hinders the reuse of process models [7].

We are working on a Universal Process Repository (UPR) that is independent of process modeling languages and is open for changes and extensions by potential users. The UPR aims to provide basic understanding of business processes and it offers a starting point for modeling business processes, by providing fundamental elements of process models.

There are several modeling languages (like YAWL [8], BPMN [9], EPC [10], IDEF0 [11] etc.) which can be used for modeling business processes. These process modeling languages have different elements and control structures [12], therefore, the specification of a business process varies from one language to another. In order to provide support for different modeling languages in the UPR, a common format for storing and sharing process models is needed, where the common format only stores fundamental elements of process models.

The XML Process Definition Language (XPDL) [13] and Business Process Modeling Ontology (BPMO) [14] are used for storing and sharing process models between different modeling languages and tools. However, the XPDL has a complex conceptual

A. Persson and J. Stirna (Eds.): PoEM 2009, LNBIP 39, pp. 176–190, 2009.

model [15] and captures detailed information about business processes. These details are redundant in the context of the scope of the UPR.

BPMO mainly intends to provide semantically rich definition of business processes, and it is focused to align the business and technical view of a process [14]. Therefore, neither XPDL nor BPMO fits the UPR's requirements for storing and sharing process models (i.e. capturing and storing only fundamental elements of process models), and an automated method for storing and retrieving process models is missing. In this paper, we propose a generic data model for storing and sharing process models between different modeling languages that only captures fundamental elements of a process. In addition, an automated method of extracting process model is given.

In section 2 of this paper, we present the research approach used to develop the generic data model for UPR. In section 3, we present the generic metamodel for a business process and the generic process description. In section 4, process description for specific languages is presented and in section 5, we present the generic data model and discuss how it can be used. Finally, we discuss and draw conclusions in section 6.

2 Research Approach

Reusability is the likelihood that artifacts can be used again with slight or no modification. The UPR intends to provide a mass of reusable process models that captures fundamental elements of a business process. For storing the fundamental elements we propose a generic data model for process description DB (a component of UPR), that can store reusable process models independent of any process modeling language. Therefore, it is likely that some language specific details of process models may not be captured.

Processes are modeled in specific languages (like BPMN, YAWL etc.) and in UPR their fundamental elements are stored in a generic format. Therefore, in order to facilitate the conversion from generic metamodel to language specific metamodel and vice versa we use a *mapping specification*. This specification is an association between elements of generic metamodel and language specific metamodels. In the remaining part of this section, we present the approaches used to develop the generic data model for process description DB (a part of the UPR). The *generic data model* consists of a *partial data model* and *mapping specification*, where *partial data model* is used to store data about processes independent of language and a *mapping specification* is used for interpretation of stored data.

Based on the definition of process model and a business process, process description is defined. However, the definition is high-level so the concrete elements of process description are not visible. Therefore, a generic metamodel is generated in order to provide concrete elements of a business processes. The generic metamodel is matched to the definition of process description to form a generic process description that is further used to develop the *partial data model*.

In order to define the *mapping specification* (another part of the generic data model), the elements of generic metamodel are matched to the elements of modeling languages. Based on the matching and the generic process description we define a language specific process description that further contributes to defining the *mapping*

specification as shown in figure 1. Each label in figure 1 is of the form *'label (number)'*, where *label* is the name of the step and *number* represents the corresponding section of the paper in which the step is discussed. The boxes with the borders represent the steps, while the ones without borders represent the output of preceding step and input to the next step.

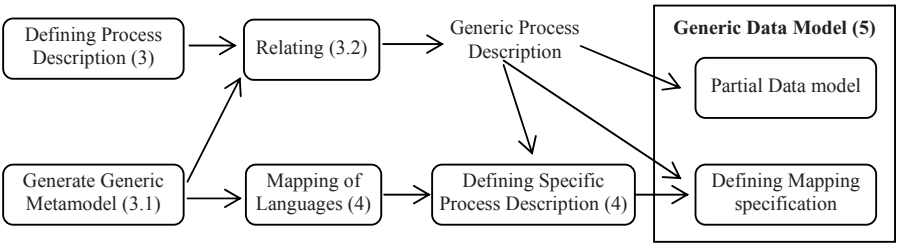

Fig. 1. Research Approach for Designing Generic Data Model for Process Description DB

3 Process Description for Storing Business Processes

In the Universal Process Repository, process definitions exist at two levels, *user level* and *repository level*. The *user level* is a higher level at which a business process is viewed as a process model. At this level a business process is modeled by using graphical constructs of a process modeling language, like BPMN, EPC, YAWL etc. However, the process model is not directly stored in the repository. The *repository level* is a lower level at which a business process is stored as a *process description*. The process description is not directly accessible to users of the repository, but it can be modified by changing its respective process model at the user level because a process description in the repository is derived from a process model. Figure 2 shows the relationship between the two levels.

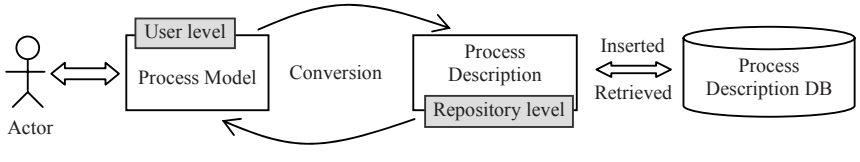

Fig. 2. Relationship between Process Model and Process Description

The benefits of storing process descriptions (non-graphical form) over process models (graphical form) includes, a) efficient retrieval and manipulation of the stored processes, b) easy and flexible way to control process models, because access to parts of a process model can be defined. Definitions of a process model and a business process are as follows [16, 17]:

A process model is defined as

"A graphical depiction of a business process detailing the arrangement of task interdependency, controls, and allocated resources."

A business process is

"A collection of related, structured activities or tasks that produce a specific service or product (serve a particular goal) for a particular customer or customers."

From the two definitions the following can be observed about a business process, a) it consists of activities (may also be called as tasks), b) activities are related with each other and dependences may exist between them, c) activities are related to resources, and d) activities are related to agents. Therefore, from these observations we define the specification of a process description as:

$$PD = \{Elements, Control\text{-}flow, Process\ Logic\} \quad \text{---------- (equation 1)}$$

Elements of a process model are the fundamental units of a process model, i.e. activities, resources and agents. *Control-flows* are the possible structures (control structures) between multiple elements in a process model e.g. sequence, AND, OR etc. *Process logic* is a logical association/binding between elements of a process model and it is defined between two elements of a process model e.g. a sequence between two activities, or resource allocation to an activity.

3.1 Generic Metamodel for Business Processes

The process description defined in the preceding section is very abstract so the concrete elements of process description are not visible in equation 1. Therefore, a generic metamodel is required to provide concrete elements of business processes. These concrete elements are used (in section 3.2) for extending the equation 1 to form the generic process description.

Process model represents information about what is done, where and when it is done, who does it, how, why, and who is dependent on its being done. In order to capture these business concepts, Curtis [18] has presented four perspectives of a business process. Here, we use the four perspectives, as defined in Curtis framework, to develop a generic metamodel for a business process shown below in figure 3.

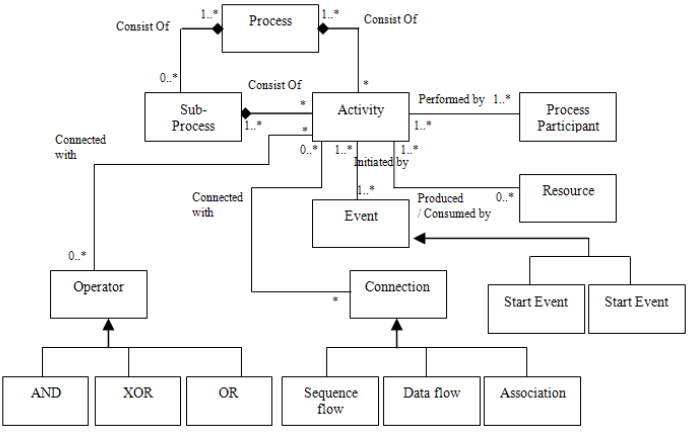

Fig. 3. Generic Metamodel for Business Processes

The four perspectives are functional, behavioral, organization and informational perspective.

Functional Perspective. The functional perspective defines the elements which are being performed. This perspective represents *'what'* elements (activities) of a process model are performed [18].

It is established that a process consists of a set of *activities*, and a process may consist of *sub-processes* [19] which also consist of activities. The execution of a business process is initiated by an event called *start event* and terminated by an event called *end event*. A *sub-process* is a part of a process that can exist independently, whereas an *activity* is a concrete instance of a task that is obtained while executing a particular case of a business process [20].

Behavioral Perspective. This perspective defines the order in which activities are executed and the point(s) where they are be executed. The behavioral perspective represents '*when'* and '*how'* activities of a process model are performed [18]. It gives dependencies between activities and how they are to be executed. These dependencies are called control flow

Control flow is a relationship between two or more activities and it can be of two types [19], *operators* and *connection*. *Operator* is a node that is used to split or join more than two elements and can either be an XOR, OR or AND. C*onnection* is an edge that connects two activities, an activity with a participant, an activity with an events, or an activity with a resource. There are three types of connections: sequence flow, data flow and association.

Organizational Perspective. This perspective defines the organizational units where business activities are performed and the involved agent. The organizational perspective represents *'where'* and *'by whom*' business activities are realized.

It is established that an activity can consume, produce, or transfer a resource between *participants* [21]. *Participants* are the entities that can execute a task to take/transfer control of a resource [22] and they could be an organizational unit or agent [19]. Organizational units may have predefined duty-descriptions called roles, that are assigned to agents. Agent can be a person, software, machine or a service.

Informational Perspective. This perspective defines *resources* that are consumed or produced during the execution of a business activity. The resources are the information elements and the documents in which these elements are stored. Information elements are the messages or data about the activity e.g. messages about output of an activity that can affect execution of the following activity.

3.2 A Generic Process Description

In this section, we relate the elements of generic metamodel and the definition of process description (given in equation 1) to form a generic process description. The process description becomes.

$$PD = \{Elements, Control\text{-}flow, Process\ Logic\}$$

Where, Elements = {start-event, activity, participant, resource, end-event}
Control flow = {operator, connection}
; connection = sequence flow, dataflow, association
Process logic = *{(start-event, activity, connection) (activity, activity, connection), (participant, activity, connection) (task, operator, connection) (operator, activity, connection)* (task, stop event, connection) (resource, activity, connection) (activity, resource, connection) (operator, operator, connection)}

Equation 2

And connection is a combination of the *type* and *label* of the connection.

4 Matching Process Modeling Languages to the Generic Metamodel

In this section we use the concepts of generic metamodel and the concepts of specific languages for matching the concepts. Later, the matching results and equation 2 (Definition of Process Description) are used for producing language specific process descriptions.

For matching concepts, we have selected four languages, Activity Diagram (AD) [23], BPMN [9], EPC [10], YAWL [8]. *Activity Diagram*, because it is a form of UML diagrams that is widely accepted and used in industry. *BPMN*, because it is a standard language, developed by OMG [24]. *YAWL*, because it is one of the most researched process modeling language. *EPC*, because it has been initiated by industry leaders in BPM, and it is supported by several tools like Oryx, ARIS etc.

Table 1. Matching Concepts of Generic Metamodel and Process Modeling Languages

Generic Metamodel	***Activity Diagram***	***BPMN***	***EPC***	***YAWL***
Subprocess		Subprocess	Sub process	
Activity	Activity	Task	Function	Task
Event				
Start Event	Initial node	Start Event	Pre-activity Event	Input condition on activity
End Event	Final node (process)	End event	Post-activity Event	Output condition on activity
Control Flow				
Operator	Forknode, join, decision, merge	AND/OR/XOR, Complex	XOR, AND, OR connector	AND, XOR, OR split & join
Connection	Control flow, Object flow	Sequence flow, association, message flow	Control flow	No formal name of the construct
Participant		Pool	Org. Unit, Org Role	
Resource				
Informational resource	Object Node	Data objects	Information objects	

Table 1 shows the results of matching concepts between generic metamodel and specific languages. The purpose of activity in generic metamodel is the same as the purpose of *activity* in AD, *function* in EPC and *task* in YAWL/BPMN. Therefore, the activity concept in generic metamodel can be matched to activity, function and task in AD, EPC and BPMN/YAWL. Similarly, the concepts in the generic metamodel are matched to specific languages as shown in table 1.

By using the matching (from table 1) and process description (from equation 2), language specific process description can be produced. This is done by considering equation 2 as a template and filling values from the matching table 1. The language specific process description for BPMN is as follows.

Language Specific Process Description: BPMN

Elements = {start event, tasks, pools, dataobject, end event}
Operators = { AND, OR, XOR}
Connection = {sequence flow-ID, message flow-ID, association-ID} ; ID is unique identifier
Process logic = {(start-event, task, connection) (task, task, connection), (pool, task, connection) (task, operator, connection) (operator, task, connection) (dataobject, task, connection) (task, dataobject, connection) (task, stop event, connection) (operator, operator, connection)}

Similarly, process description for YAWL, Activity Diagram and EPC can also be defined.

5 Generic Data Model for Process Description DB

The generic data model is a logical data model that defines structure and content of a generic process description in the UPR. The generic data model is used for capturing and storing fundamental elements of process models in the repository. The generic data model consists of a *partial data model* and *mapping specification*, where the *partial data model* is used to store data about processes independent of modeling language and *mapping specification* is used for interpretation of the stored data.

The generic data model is generated from a generic process description as defined in equation 2 of section 3.2. The model consists of entities, attributes and relationship between entities as shown below in figure 4. The entities are derived from the elements and control flow and the relationships are derived from the process logic.

Event. In the generic process description there is a start and an end event. In order to capture the information about start and end event an entity (labeled *Event*) is included in the data model. The event consists of three attributes, (a) EventID that describes a unique identification of an event in the table (b) EventName that describes the name of an event (i.e. Illness Occur), and (c) EventType which specifies a type of a particular event and it can be either a start event or end event.

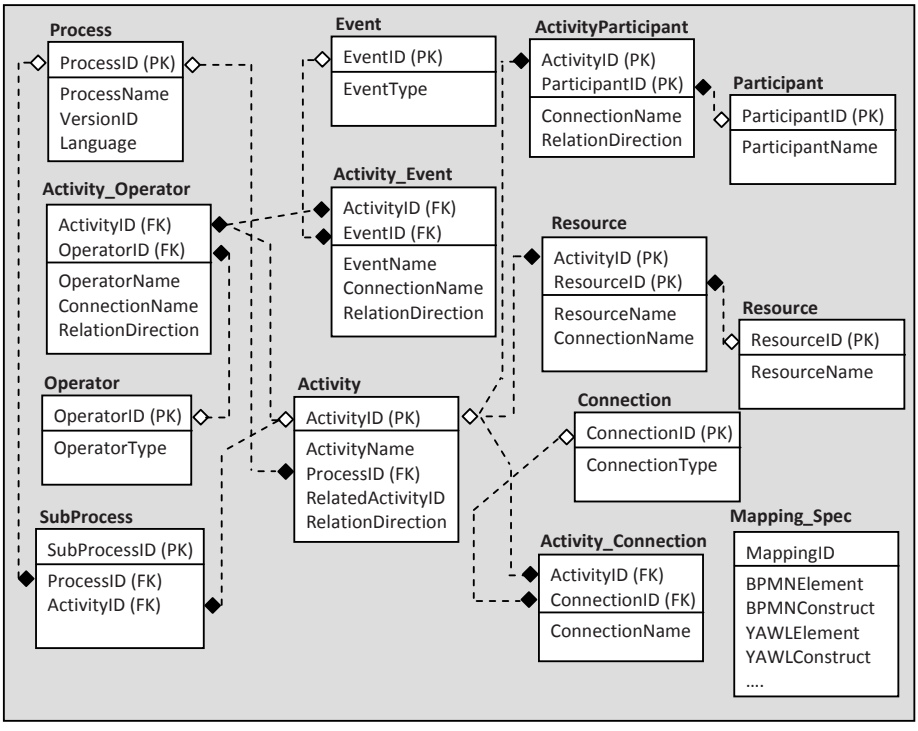

Fig. 4. Relationship between Process Model and Process Description

Activity. In the generic process description we have activity as an element, therefore, in order to capture the information about activities we include an entity labeled as *Activity* in the data model. The activity consists of the following attributes (a) ActivityID that specifies a unique identifier for each record in the table. (b) ActvityName describes the name of an activity (i.e. Prepare Order) and (c) ProcessID as a foreign Key for the entity Process.

Similarly, the participant and resource (as elements) and operator and connection (as controlflows) in the generic process description are represented by the entities participant, resource, operator and connection in the data model. The entities ActivityEvent, ActivityOperator and ActivityConnection are generated from many to many relationships between the three entities (Event, Operator and Connection) and the Activity.

In order to facilitate the use of the data model we introduce a mapping specification as a disjoint entity in the generic data model. The *mapping specification* entity is derived from the generic process description and language specific process description. It consists of elements of a generic process description, their corresponding elements in the specific process description and their associated construct.

5.1 On Using the Generic Data Model

In this section we describe how the generic data model can be used in the Universal Process Repository for storing and extracting process models in the form of process description. Once the mechanism of storing and extracting process models is established, the process models can be used to share and reuse process model.

From a user perspective, the user can interact with a modeling editor to model processes in one of the process modeling languages. At the backend of the editor we have a mechanism for extracting information from a process model and to store it in the Process Description DB. Similarly, a mechanism for the retrieval of data and its conversion to a process model is also used. In the remaining part of this section, we present the mechanism (in the form of pseudo code) for retrieving data from Process Description DB and its conversion to a process model.

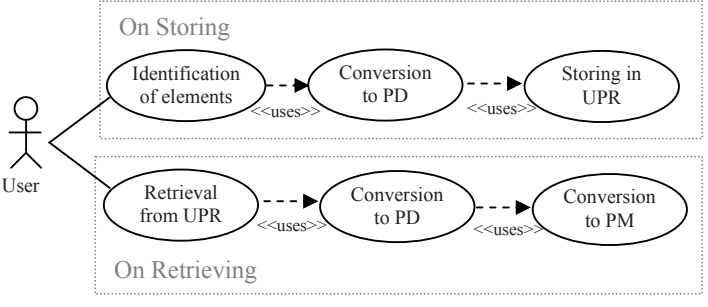

Fig. 5. Relationship between Process Model and Process Description

The algorithm for storing process model is similar to the one used for retrieval, therefore, to avoid redundancy it is not presented in this paper. However, figure 5 shows high-level use cases for storing and retrieving process models.

Pseudo Code of Extraction Algorithm. The algorithm shows the retrieval of a process with the assumptions that the process is identified and will be retrieved in the language in which it was stored. Primarily, it is a three phase algorithm, a) data retrieval from process description DB, b) generation of process description (PD) from the retrieved data, c) conversion from PD to BPMN process model. As soon as the retrieval takes place from DB, the process description is generated. However, for clarity these phases are separated from each other.

Input: identified process description
Output: a process model in BPMN
Create alias for the data model
{ activity → task,
resource → dataobject
participant → pool }

a) Retrieval from Process Description DB:

Use *event* and *process* table

{ Identify *(start event)*
Identify *(end event)* }

Use *task*, *event* and *connection* table

{ Identify (the *task/s* related to *start event*)
& *connection* from *start event* to each *task* }

For each related *task* **do**

{ Identify (*task* related to *task*) & *connection* from *task* to *task*
Identify (*pool* related to *task*) & *connection* from *pool* to *task* Identify (*data object* related to *task*) &
connection from *data-object* to *task*
Identify (*task* related to *operator*) &
connection from *task* to *operator* }
While *task* is not null

For each *operator* **do**

{ Identify (*operator* related to *task*) &
connection from *operator* to *task*
Identify (*operator* related to *operator*) &
connection from *operator* to *operator* }
While *operator* is not null

b) Conversion to Process Description:

The template of the process description for BPMN is given below and is used to structure the data before it is converted to process model. The template given below is instantiated.

Elements = {*start event, tasks, pools, dataobject, end event*}

Operators = { *AND-ID, OR-ID, XOR-ID*} ; where
ID is used to uniquely identify each operator

Connection = {*sequence flow-ID, message flow-ID, association-ID*} ; ID is unique identifier

Process logic = {*(start-event, task, connection) (task, task, connection), (pool, task, connection) (task, operator, connection) (operator, task, connection) (dataobject, task, connection) (task, dataobject, connection) (operator, operator, connection) (task, stop event, connection)*}

c) Conversion to Process Model:

The structured form of data is converted to a graphical form (called process model). Input to this step is each instance of process logic and draw function is used for modelling a process on-the-fly. A process model is completed when all instances of the process logic are passed to the Draw function.

Draw (X, Y, Z) /*where X is the first element, Y is follow up element and Z is the connection between X and Y.

{

If (X & Y= *'NotNull'* and *not represented already*)
{ Find (*construct* of X from *mapping specification*)
& (*construct* of Y from *mapping specification*)
& (*construct* of Z from *mapping specification*)

```
                Place construct X and Y and link them by Z.
                And Add Label X, Y, Z
              }
        Else if (X is represented already)
          {     Find (construct of Y from mapping specification)
          &     (construct of Z from mapping specification)
                Place construct of Y and link X,Y by Z
                And label Y, Z.
          }
        Else Get next instance and continue.
}
```

5.2 Generating Process Model from Process Description DB: An Example

In order to exemplify the use of the proposed algorithm and the generic data model, we consider a process model from the Swedish healthcare sector, as shown below in figure 6. For the process model we presented how the data is stored in the process description DB and from which we reproduce the same model. The purpose of this exercise is to exemplify the use of the algorithm and identify the information lost during storing and retrieving process model. This is done by comparing the original and the retrieved process model.

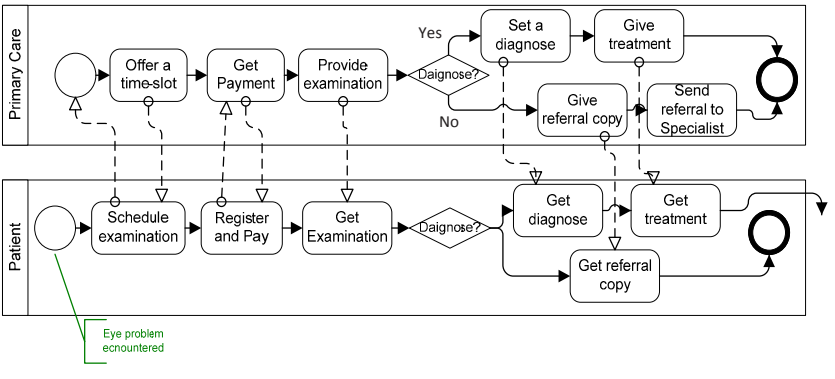

Fig. 6. Example Eyecare Referral Process

The instances of the tables become as follows, where each tuple is separate by ";":
Process = *1, eye care referral process, 1, BPMN;*
Event = *1, start event; 2, end event*
Activity = *1, offer a time-slot, 1, 2; 2, get payment, 1, 3;…*
ActivityEvent = *1, 1, receive a call, null, 1; 1, 2, eye problem encountered, null, 1; 10, 2, null, null, 1;…*
Operator = *1, OR;*
ActivityOperator = *5, 1, diagnose, null, 1; 7, 1, diagnose, yes, 1; …*
Connection = *1, sequenceflow; 2, dataflow ;*
ActivityConnection = 1, 1, null, 1; 1, 2, null, 0;…
Participant = 1, primary care; 2 patient;
ActivityParticipant = 1, 1, null, 1; 2, 2, null, 1;…

The start event, tasks, pools, dataobjects, operators and connection are given above, so they are not repeated here. Whereas the process logic is as follows.

(start-event, task, connection) = (*receive a call, 1, sequenceflow-null*) // where in sequenceflow-null, sequenceflow is the type of flow null is the label. 1 is the taskID.
(task, task, connection) = (*1, 2, sequenceflow-null*); (*2, 4, messageflow-null*)…
(pool, task, connection) = *(patient, 2, messageflow-null); (primaycare, 1, messageflow-null)…*
(task, operator, connection) = (*5, OR-diagnose?, sequenceflow-null*)
(operator, task, connection) = *(OR-diagnose?, 7, sequenceflow-yes); (OR-diagnose?, 8, sequenceflow-no)…*
(task, stop event, connection) = *(12, A, sequenceflow-null); (10, B, sequenceflow-null)…*

By using the draw function of the algorithm, a process model can be developed. Figure 7 shows the step by step development of a process model. When the first instance of process logic of the type *(start-event, task, connection)* is passed to the draw function, *start event* its related *task* and *connection* between them (start event and task) is created as shown in step 1 in figure 7. On passing *(task, task, connection)* the second step is executed.

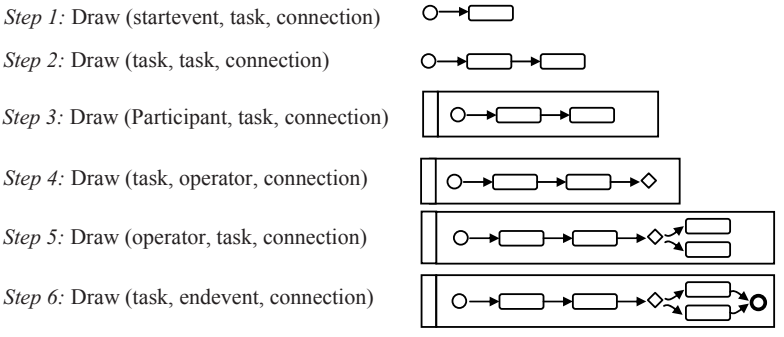

Fig. 7. Step by Step Process Model Generation

Figure 8 shows the process model produced by the draw function, when all instances of the process logic (of health care process) are passed to it.

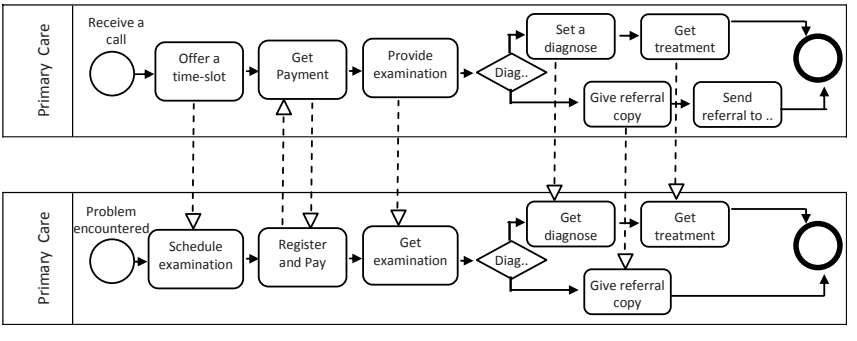

Fig. 8. Example Eyecare Referral Process: On Retrieval

6 Discussion and Conclusions

Modeling of processes is a complex and time consuming task which can be simplified by reuse of process models. A repository is, therefore, necessary to store and manage process model. However, the mass of process models available in repositories cannot be reused because, process models are either domain specific processes or the repositories are not publically open for change and growth [7]. Furthermore, most repositories are proprietary and not extensible. To overcome these limitations, work on a Universal Process Repository is in-progress.

In this paper, a generic data model of process description DB (a component of UPR) is proposed that is capable of storing fundamental information of processes independent of any language. For developing this data model we start from the definition of business process and process model and follow step-by-step procedure to generate the logical data model.

Users of the repository interact with the process editor, and model their processes which are automatically stored in the process description DB in non-graphical form. Similarly, the retrieval process is also automated and transparent to users. Data from Process description DB is retrieved and converted to process models with the help of the algorithm presented in section 5.2. Also, we applied the approach on an eye care referral process to demonstrate the applicability of the proposed approach.

From the study it can be concluded that, a) the generic data model is capable of storing fundamental elements of a process model in a format that is independent of any process modeling language, b) the automated way of storing and retrieving process models can facilitate users in storing and retrieving process models without any additional effort and c) once stored, the process models can be shared and reused. Only the fundamental elements of process models are captured, therefore, only basic understanding of a process can be provided that can be used as a starting point for process modeling. Thus, some advanced features that are specific for each language may not be captured by the logical data model. Examples of these features are token and composite task in YAWL, sending signal to outside and activity partition in AD, and intermediate event in BPMN. This may in some cases result in losses during translation, i.e. the translation will not preserve the semantics of the original model. Such losses may be unacceptable for certain applications but are less problematic for UPR, as we only intend to provide basic understanding and fundamentals of process models.

Transforming process descriptions between different languages and elicitation of guidelines for populating the repository are some of the future research directions.

References

[1] Rodrigues, J.A., Souza, J.M., Zimbrao, G., Xexeo, G., Neves, E., Pinheiro, W.A.: A P2P Approach for Business Process Modelling and Reuse. In: Eder, J., Dustdar, S. (eds.) BPM Workshops 2006. LNCS, vol. 4103, pp. 297–307. Springer, Heidelberg (2006)

[2] Hornung, T., Koschmider, A., Oberweis, A.: A Recommender System for Business Process Models. In: Proceedings of the 17th Annual Workshop on Information Technology and Systems (WITS 2007), Montreal, Canada (2007)

[3] Ma, Z., Leymann, F.: A Lifecycle Model for Using Process Fragment in Business Process Modeling. In: Proceedings of the 9th Workshop on Business Process Modeling, Development and Support (BPMDS 2008), in Conjunction with CAiSE 2008 Conference, France, (2008)
[4] IBM Process Repository, http://publib.boulder.ibm.com/infocenter/wchelp/v5r6m1/index.jsp?topic=/com.ibm.commerce.business_process.doc/concepts/processPrice_order.htm (last accessed on May 20, 2008)
[5] Business Process Execution Language Repository, http://www.alphaworks.ibm.com/tech/bpelrepository?open&S_TACT=105AGX59&S_CMP=GR&ca=dgr-eclpsw03awbpelrepository (last accessed on May 20, 2008)
[6] SAP Business Map, http://help.sap.com/saphelp_sm40/helpdata/EN/5e/c8145e3a9d9340913099159d80fc87/frameset.htm (last accessed on May 20, 2008)
[7] Shahzad, K., Andersson, B., Bergholtz, M., Edirisuriya, A., Illayperuma, T., Jayaweera, P., Johannesson, P.: Elicitation of Requirements for a Business process Model Repository. In: Ardagna, D., et al. (eds.) BPM 2008 Workshops. LNBIP, vol. 17, pp. 42–53. Springer, Heidelberg (2008)
[8] Aalst, W.M.P., Hofstede, A.H.M.: YAWL: Yet Another Workflow Language. QUT Technical Report, FIT-TR-2002-06, Queensland University of Technology, Brisbane (2002)
[9] BPMN Specification Release, Object Management Group (OMG), http://www.omg.org/spec/BPMN/1.2/ (last accessed on June 30, 2009)
[10] Aalst, W.M.P.: Formalization and Verification of Event-Driven Process Chains. Information and Software Technology 41(10), 639–650 (1999)
[11] IDEF0, FIPS Publication 183, Computer Systems Laboratory, National Institute of Standards and Technology (NIST), http://www.idef.com/idef0.html
[12] Lu, R., Sadiq, S.: A Survey of Comparative Business Process Modeling Approaches. In: Abramowicz, W. (ed.) BIS 2007. LNCS, vol. 4439, pp. 82–94. Springer, Heidelberg (2007)
[13] XML Process Definition Language (XPDL), Standard by Workflow Management Coalition (WfMC), http://xml.coverpages.org/XPDL20010522.pdf
[14] Cabral, L., Norton, B., Domingue, J.: The Business Process Modeling Ontology. In: Proceedings of the International Workshop on Semantic Business Process Management, collocated with ESWC 2009, Crete, Greece (2009)
[15] Chinosi, M.: Representing Business Processes: Conceptual Model and Design Methodology. PhD Thesis, Department of Computer Science, Università degli studi dell'Insubria, Italy (2009)
[16] http://www.cecausa.com/business_process_glossary.htm Process model definition
[17] Johannesson, P., Andersson, B., Bergholtx, M., Weigand, H.: Enterprise Modelling for Value Based Service Analysis. In: Stirna, J., Persson, A. (eds.) POEM 2008. LNBIP, vol. 15, pp. 153–167. Springer, Heidelberg (2008)
[18] Curtis, B., Kellner, M.I., Over, J.: Process Modeling. Communications of ACM 35(9), 75–90 (1992)
[19] List, B., Korherr, B.: An evaluation of conceptual business process modelling languages. In: Proceedings of ACM Symposium on Applied Computing (SAC 2006), Dijon, pp. 1532–1539 (2006)

[20] Axenath, B., Kindler, E., Rubin, V.: An Open and Formalism Independent Meta-Model for Business Processes. In: Proceedings of the BPRM 2005, in conjunction with 3rd BPM 2005, Nancy, France, pp. 45–59 (2005)
[21] Edirisuriya, A., Johannesson, P.: On the Alignment of Business Models and Process Models. In: Ardagna, D., et al. (eds.) BPM 2008 Workshops. LNBIP, vol. 17, pp. 68–79. Springer, Heidelberg (2008)
[22] Johannesson, P., Andersson, B., Bergholtz, M., Weigand, H.: Enterprise Modelling for Value Based Services Analysis. In: Stirna, J., Persson, A. (eds.) POEM 2008. LNBIP, vol. 15, pp. 153–167. Springer, Heidelberg (2008)
[23] Dumas, M., ter Hofstede, A.H.M.: UML Activity Diagrams as a Workflow Specification Language. In: Gogolla, M., Kobryn, C. (eds.) UML 2001. LNCS, vol. 2185, pp. 76–90. Springer, Heidelberg (2001)
[24] Object Management Group (OMG), `http://www.omg.org/`
[25] BPMN Specification Release, Object Management Group (OMG), `http://www.omg.org/spec/BPMN/1.2/` (last accessed on June 30, 2009)