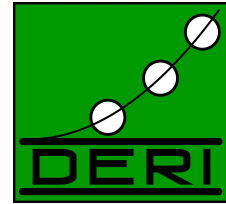


DERI – DIGITAL ENTERPRISE RESEARCH INSTITUTE



ANALYSIS AND MINING OF  
ONTOLOGICAL PROCESS MODEL  
INSTANCES

Armin Haller      Walid Gaaloul  
Mateusz Marmolowski

DERI TECHNICAL REPORT 2008-03-28

MARCH 2008

DERI – DIGITAL ENTERPRISE RESEARCH INSTITUTE

**DERI Galway**  
IDA Business Park  
Galway, Ireland  
[www.deri.ie](http://www.deri.ie)



## DERI TECHNICAL REPORT

DERI TECHNICAL REPORT 2008-03-28, MARCH 2008

# oXPDL: A PROCESS MODEL EXCHANGE ONTOLOGY

Armin Haller<sup>1</sup>, Walid Gaaloul<sup>1</sup>, Mateusz Marmolowski<sup>1</sup>

**Abstract.** The application of ontologies to Business Process Management has been introduced to improve the level of automation in the execution of processes. Ontologies aid in the description of the process model as well as in their exchanged data. This paper presents an ontological model for workflow logs, namely oXPDL+. The model builds upon a process interchange ontology based on the standardised XML Process Definition Language (XPDL). oXPDL+ can be used to not only exchange process models, but also workflow logs representing instances of such process models. We present a mapping architecture and implementation to populate a knowledge base with ontologically described process models and workflow logs. By defining a set of ordering relations in the ontological model, it is possible to analyse the model based on a combination of its static and behavioural properties. The use of semantics to model processes allows to interlink local information with knowledge defined in background ontologies, leading to significant enhancements in analysis and mining techniques.

---

<sup>1</sup>Digital Enterprise Research Institute (DERI), National University of Ireland, Galway, IDA Business Park, Lower Dangan, Galway, Ireland. E-mail: {firstname.lastname}@deri.org.

**Acknowledgements:** This material is based upon works supported by the Science Foundation Ireland under Grant No. SFI/02/CE1/I131 and No. SFI/04/BR/CS0694.

Copyright © 2008 by the authors

## 1 Introduction

The use of information technology to improve the efficiency of business processes is a major driver in current businesses. Workflow Management Systems (WfMS) were introduced in the early Nineties to offer generic modelling and enactment capabilities for structured business processes. Besides pure WfMS many other information systems have adopted workflow technology. Today, the majority of enterprise information systems are process-aware, using some mechanism to support, control, and monitor business processes [8]. Typical examples of such systems, driven by implicit or explicit process models, are *Enterprise Resource Planning* systems, *Customer Relationship Management* systems and *Groupware* collaboration systems. All these applications operate on explicit process models, high-level specifications of processes independent of their implementation in the system.

Despite ongoing standardisation efforts, no consensus exists on the representation of process models [19], although BPEL [20] gains widespread use to model the behavioural perspective of workflows. The standardised exchange language XPDL (XML Process Definition Language) [22] has been developed to provide a language to describe all aspects of a workflow and exchange process models between different applications. XPDL is currently supported by over seventy products<sup>1</sup>. However, XPDL has been built with the intention to exchange process models (types) only. Process types constitute the schema of a process, defining a collection of activities and setting out the control flow as well as data flow between them. Based on the schema, process instances are created at runtime described in workflow logs.

Since most WfMS have their own internal structure and language to define process models as described above, when trying to use workflow logs from different systems, a standardised way to represent these workflow logs is needed. There has been a first attempt to provide such a meta model for workflow logs with the introduction of MXML, a model and language used within the process mining framework ProM [7]. Although this model is a first necessary step, it lacks an explicit combination of a process model with its instances. As a result it can not be used to exchange runtime information, nor can multiple logs from different systems or logs based on different process models be uniformly analysed. Further, it is defined in XML Schema, lacking the formal explicit semantics of a Web Ontology language. XPDL is a natural candidate model to be used to exchange process instances. It offers an extensive process model and although not designed to be used as a process instance exchange model, it already incorporates some run-time properties within the definition of certain element types. Our ontological extensions to oXPDL concerned with workflow log modelling build upon MXML [6], taking advantage of the translators developed in the context of the ProM framework [7].

The paper is structured as follows. First we outline our approach and the advantages of semantic workflow logs for process mining and analysis in section 1.1. We briefly introduce the Semantic Web ontology language used throughout the paper in section 1.2 and relate our work to other approaches in section 1.3. In section 2 we present a motivating Repair process and formulate competency questions to be answered by the analysis of workflow logs. The example process is used throughout section 3 to illustrate the ontological extensions and to answer the competency questions on workflow log models in oXPDL+. Section 4 presents how semantics improve process mining techniques by leveraging them towards more advanced, adaptable and reusable analysis solutions. The conversion methodology and its tool implementation to translate XPDL process models and extracted workflow logs in MXML to oXPDL+ is described in section 5. We conclude and discuss further research directions in section 6.

---

<sup>1</sup><http://www.wfmc.org/standards/xpdl.htm>

### 1.1 Approach: ontologising and mining XPDL

We propose to extend the oXPDL model, a translation of XPDL to a Web ontology language presented in Haller *et al.* [13], with instance related properties. These extensions, namely oXPDL+, can model the events taking place during the execution of a process model. The concepts in oXPDL dealing with run-time information are based upon standard workflow log meta models such as the one described by MXML [6]. Such a formal definition of process instances in a Web ontology language such as OWL [15] or WSML [5], potentially allows information systems to exchange runtime information. Further, providing means to integrate workflow logs in a unified model, with a formal semantics and unambiguously identified objects through the use of ontologies, enables process mining across the execution traces of multiple information systems. By interlinking the process model and its instances to existing background ontologies improves the process instance semantics in a way that a workflow log does not only contain simple types, but identifies objects instead. The advantages of ontologising workflow logs on three common classes of process mining techniques can be summarised as follows:

**Process Analysis** The goal of process analysis and process monitoring is on the one hand to track the enactment of processes as they are performed and on the other hand to perform business intelligence operations after the execution of the processes. It supports business analysts to identify deviations in the processes and to identify corrective measures to re-engineer suboptimal processes. Semantic workflow logs allow to query a set of process instances on different levels of abstraction. In contrast to traditional approaches, the queries on the model and its instances are not limited to a specific proprietary model, but can be populated from multiple process aware systems. Business intelligence operations can easily be performed over multiple instances of the same or different process models. As such, by using an aggregated analysis one can examine results for resources within a process across all process instances.

**Semantic Process Mining** Process mining refers to techniques to extract process models from logs. It focuses on the automatic discovery of information from event logs without a predefined model. The importance of process mining in BPM is widely acknowledged as an important and unavoidable analysis tool to aid the (re-)design and the (re-)configuration of process models. Current process mining techniques are already quite powerful and mature. However, the analysis they provide are purely syntactic. In other words, these mining techniques are unable to reason over the concepts behind the labels in the log, thus the actual semantics behind these labels remain in the head of the business analyst who has to interpret them. Therefore, our aim is to develop “**semantic**” process mining techniques that make use of this semantic perspective. Combined with reasoners to support inferencing, the level of abstraction is raised from the syntactical level to the semantical level in order to derive new knowledge.

**Cross-organisational Conformance Checking** Conformance Checking refers to algorithms that verify if logs follow the predefined behaviour expressed in the process model. These algorithms require both, the process model and its instances. The main advantage of ontologically defined process instances and models is that it improves the interoperability between information systems. Based on our model, WfMS can be extended with import facilities for process instances. As such, conformance checking can be performed over multiple workflow logs, possibly even cross-organisational models. As such, cross-organisational workflows can be monitored in the

knowledge base, populated by each independent WfMS, to be aware of the global state of the execution at every point in time.

## 1.2 Ontology language

For the modelling of the oXPDL+ ontology and its implementation we have chosen the ontology language WSML [5] as our modelling language. This choice is based on the fact that WSML, as a Web ontology language, enables the integration of oXPDL+ with arbitrary (public) background ontologies and that a reasoning architecture is readily available to be used with the results of our conversion process (see section 5). Although we have opted to use WSML, our solution could also be implemented using another Web ontology language such as OWL [15]. In fact, we are currently working to generate RDF(s) [4] output in our conversion tool. The WSML ontology language itself is actually a family of several language variants: its constituents are based on different logical formalisms and have different levels of logical expressiveness.

## 1.3 Related work

The workflow log meta model presented in this paper is not the only attempt on a formalisation of data models for event logging. In fact, the MXML format, presented in [6] acted as input to our model and is used in the initial implementation as an interchange model to convert workflow logs from different WfMS to oXPDL+. Other examples of meta models in the context of process-aware information systems can be found in [17]. This work however focuses on an evaluation framework for workflow event logs. Further, our approach is to the best of our knowledge the first to a semantic process meta model, extending on the one hand an existing standard and comprising of an ontological process model and extensions to model occurrences. Recent work of de Medeiros *et al.* [16] tries to leverage classical process mining techniques into a semantic level by linking and replacing the discovered results to their respective concepts. The work is based on the toolset developed in the European IP project SUPER<sup>2</sup> and does not use or propose an interchange ontology based on a standard. In regard to the proposed process mining it uses existing techniques by proposing a semantically enhanced view over the mined results and assumes that event logs and models link to ontologies. While, in this paper, we propose more specific and robust semantic process mining techniques that aim to make use of the ontological annotations in the process models to infer concept instances in purely syntactic logs.

In a broader sense of related work, previous attempts in workflow mining seem to focus on control flow mining perspectives. van der Aalst *et al.* [2] present an exhaustive survey of relevant techniques and approaches. Prior art more closely to the semantic analysis techniques presented in this paper exist in the field of Business process intelligence limited however to estimating deadline expirations and exception prediction [10]. They describe a BPI tool set on top of HPs Process Manager is described. The BPI tools set includes a so-called “BPI Process Mining Engine”. It supports business and IT users in managing process execution quality by providing several features, such as analysis, prediction, monitoring, control, and optimisation. However, these above mentioned works do not discuss business process semantic aspects nor do they address the issue of business process log standardisation.

Data warehouses to store workflow logs were proposed in the scientific literature [9, 18]. These data warehouses simplify and accelerate the requests necessary for workflow mining techniques.

---

<sup>2</sup>[www.ip-super.org](http://www.ip-super.org)

Any information system using transactional systems such as ERP, CRM, or WfMS offers this information in a certain form. Currently, most of WfMS log all events occurring during process execution [2]. However, these logging facilities are rather proprietary and focuses on Database functionalities as a repository rather than a standardized knowledge management tool for mining and analysis facilities.

## 2 Motivating Example

In this section we present an exemplary repair process, a process similar to a reference model also used within the ProM framework [7]. The process is later used to illustrate our ontological model in section 3. Further, a set of generalised competency question is formulated which will be answered in the remainder of the paper.

Figure 1 shows an electronic device repair process, first represented as a BPMN diagram and second serialised in an oXPDL instance. The activity ordering in the process is as follows. The company dealing with the repair process accepts three different types of electronic devices for repair (ED1, ED2, ED3). The process starts by registering the part identification number of the electronic device sent by a customer. After registration, the electronic device is sent to the *Problem Detection* department to analyse the defect and categorise the malfunction. Once the problem is identified, the electronic device is sent to the *Repair* department and the customer is informed about the problem and the expected costs of the repair. The *Repair* department has two teams. One team of junior technicians, able to fix simple problems, and a senior technician and his team who is responsible to fix complex defects. However, some of the defect categories can be repaired by both teams. Once a repair employee finishes the repair, this device is sent to the *Quality Assurance* department. It is analysed if the defect was properly fixed. If the device is not working properly, it is returned to the *Repair* department with a specification of the problem. If the electric device is indeed repaired, the case is archived and the electric device is sent to the customer. To save on throughput time, the company only tries one iteration of the repair cycle. If the defect can not be fixed then, the case is archived and a brand new device is sent to the customer.

**Competency questions** Our ontological version of XPDL may be used to answer on a set of process instances and their respective process models, a defined set of competency questions [24], listed below. Throughout the rest of the paper, we will present queries on the oXPDL+ ontology that answer these questions for our motivating example.

1. How many times has the “Repair” process being suspended?
2. What products are handled in the “Analyse Defect” task?
3. What number of “Analyse Defect” tasks is handled within two hours?
4. Which tasks precedes the “Archive Repair” task in instances of the “Repair” process?
5. What is the most frequent path in the “Repair” process?
6. Are there any multiple choice tasks in the “Repair” process?
7. Who works on the “Simple Repair” task in instances of the “Repair” process?
8. What department is concerned with the “Analyse Defect” task in the “Repair” process?
9. How many people are actually involved in a repair case?

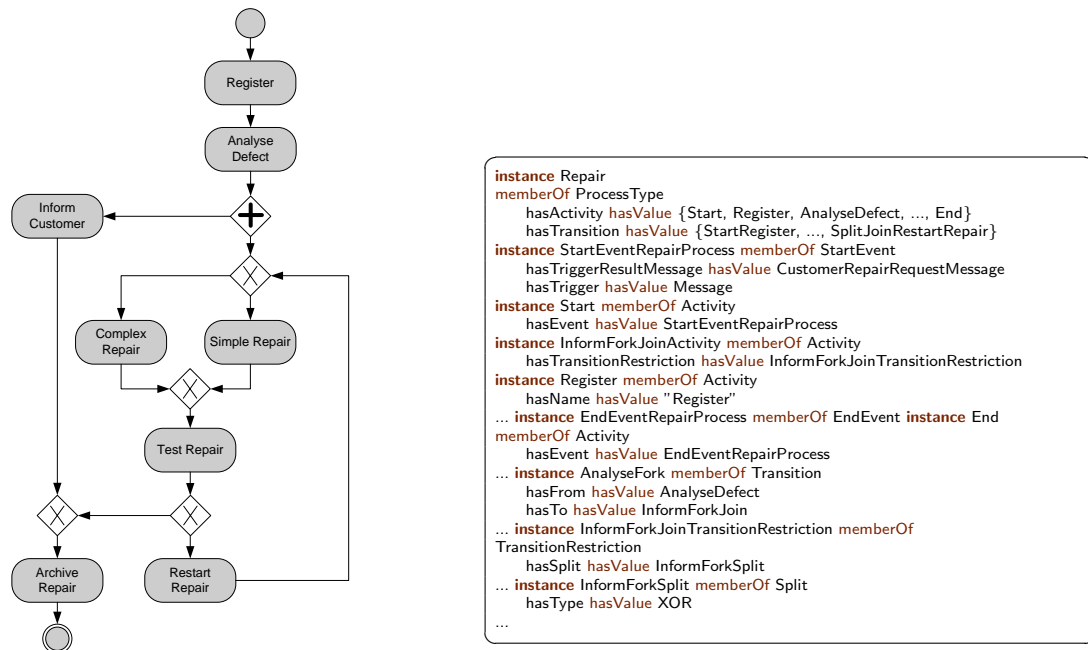


Figure 1: BPMN diagram of a Repair process (*left*) and snippets of its serialisation in oXPDL (*right*)

### 3 oXPDL+ ontology

In this section we present extensions to oXPDL to model process instances. Through the use of both, the process model in oXPDL and the process instances modelled in oXPDL+, logical relations can be established between the two. The modelling of the two hierarchical layers of model types and model instances within the ontological model is handled as follows:

**Concepts versus instances** *Ontological concepts* are used to represent a type of an entity, i.e. an *Activity* in XPD. The translation of an XML instance according to the XPD schema to its ontological representation creates *instances* for each of these generic model type concepts, e.g. “Register” in our motivating example represents an instance of the *Activity* concept.

**Occurrences versus instances** *Occurrence concepts*, denoted by the keyword “occurrence”, model instances of a workflow model, e.g. “Repair210920071123” represents a “Repair” process instance that started on the 21/09/2007 at 11:23.

This approach allows one to model an XPD document as an instantiation of oXPDL concepts, while instance information is represented as instances of *occurrence concepts* from oXPDL+. Figure 2 shows the meta model of these occurrence concepts in oXPDL+. The complete ontology consists of about 125 concepts. For space considerations, however, we refer to Haller *et al.* [13] for full details on the process model of oXPDL.

The remainder of this section describes the occurrence concepts of oXPDL+, logically structured according to the classification of workflow models in standard literature [1, 14] into three dimension, a functional dimension, a behavioural dimension and a resource dimension. In each section we

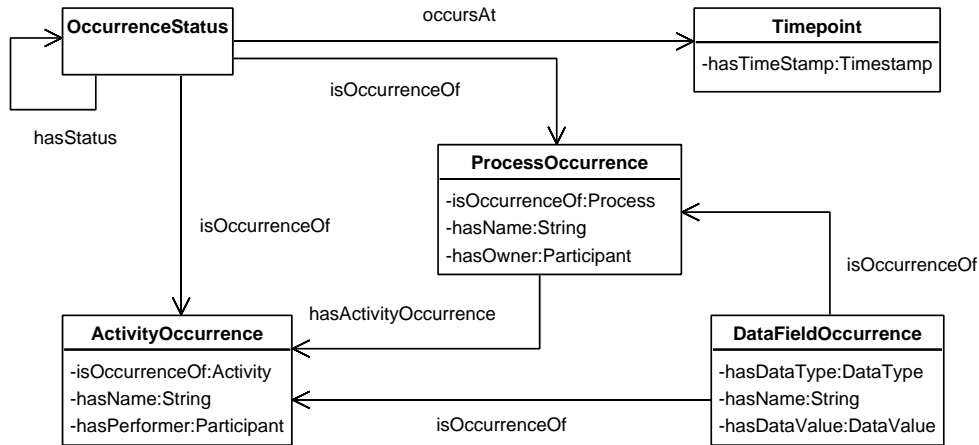


Figure 2: oXPDL+ Meta Model

briefly describe the oXPDL model type concepts, followed by a discussion of the concepts dealing with process instance information in oXPDL+. For each dimension we show how process analysis techniques can benefit from an ontological model. Queries on the model are illustrated throughout each section on the example introduced in section 2 to answer the competency questions raised in the same section.

### 3.1 Functional dimension constituents

The functional aspect provides modelling facilities to define generic properties of workflow models. It can be seen as a meta-concept concerned with elements which are shared between multiple processes, such as package information, type declarations or partner links.

The majority of functional properties in the oXPDL model are referenced through the *Package-Header* concept, defining the hierarchical layering of workflows, the type declarations, applications and artifacts shared between process models, and the properties related to the graphical layout to represent a BPMN model or a proprietary graph-based model.

Additional information required to model process instances in oXPDL+ are mainly concerned with status information, data and performative aspects of a process. Most importantly, a *OccurrenceStatus* concept is used to model the execution state of the entire process or parts of it. The status information (i.e. occurrence states such as *active*, *suspended*, *resumed*, *cancelled*, *aborted*, *assigned* etc.) can be used to determine the current state of execution (if queried at runtime) or to analyse the execution results of individual tasks or the entire process after completion of the process. In contrast to traditional workflow logs, queries on the execution results of a task or process in the ontological model can also range over corresponding relationships (e.g. *ProductIdentification* information can refer to background ontologies as shown in listing 2). Competency question #1 (“How many times has the repair process being suspended?”) can be answered with a query ranging over status properties shown in listing 1.

```
?a[isOccurrenceOf hasValue Repair] memberOf ProcessOccurrence and
?a[hasStatus hasValue suspended] memberOf ProcessOccurrence.
```

Listing 1: Answering competency question #1

Whereas the process model defines the data types and therefore the constraints on the interpretation of its data, the values and objects according to their type definition have to be captured at any given point in time during the execution of a process. In WfMSs this information is stored in the database, but commonly not explicitly represented in the workflow log. In contrast, in a semantic log based on oXPDL+, data values are represented in the model as a self-sufficient concept. As such, changes to the data value during the execution of a task or process can be captured. Listing 2 shows the resource “gtin#ProductIdentification” modelled as a data type required by the “Simple Repair” process and referenced by its Global Trade Identification Number in the namespace of a *GTIN* ontology. The second part of the listing shows instances of the process model from a workflow log, referencing actual product identification numbers.

```

/* Process Model instances */ instance AnalyseDefect memberOf
Activity
  hasName hasValue "Analyse Defect"
  hasDataField hasValue gtin#ProductIdentification

instance gtin#ProductIdentification memberOf DataField
  hasDataType hasValue .string

/* Workflow log instances */ instance AnalyseDefect211020071230
memberOf ActivityOccurrence
  isOccurrenceOf hasValue AnalyseDefect
  hasDataFieldOccurrence hasValue .614141000883

instance AnalyseDefect301120071532 memberOf ActivityOccurrence
  isOccurrenceOf hasValue AnalyseDefect
  hasDataFieldOccurrence hasValue .614141000884

instance .614141000883 memberOf DataFieldOccurrence
  isOccurrenceOf hasValue gtin#ProductIdentification
  hasDataValue hasValue gtin#_614141000883
  hasName hasValue "Electronic Device 883"

instance .614141000884 memberOf DataFieldOccurrence
  isOccurrenceOf hasValue gtin#ProductIdentification
  hasDataValue hasValue gtin#_614141000884
  hasName hasValue "Electronic Device 884"

instance completed211020071532 memberOf OccurrenceStatus
  isOccurrenceOf hasValue AnalyseDefect211020071230
  hasStatus hasValue completed
  occursAt hasValue .1192980740

```

Listing 2: Workflow snippet including Data

Having this data, one can query the knowledge base for competency question #2 (“What products are handled in the “Analyse Defect” task?”) as follows.

```

?a[isOccurrenceOf hasValue AnalyseDefect] memberOf
ActivityOccurrence and ?b[isOccurrenceOf hasValue ?a] memberOf
DataFieldOccurrence and ?b[hasName hasValue ?c] memberOf
DataFieldOccurrence.

```

Listing 3: Answering competency question #2

Time is a pivotal property of traditional workflow logs. All tuples in a workflow log have a time property attached and the log analysis offered by WfMSs is usually confined to the ability of retrieving the number of process instances completed in a given time period and their average execution

time [3]. Process and activity instances are in exactly one specific state (modelled as *OccurrenceStatus* as shown in listing 2) at any given point in time. For this reason, time is an attribute to the *OccurrenceStatus* concept, which itself references the *ProcessOccurrences* and *ActivityOccurrence* concept. As such, every new state is modelled as an instance of the *OccurrenceStatus* concept with the time of change attached. Transitions between process and activity occurrences are modelled as relations over the respective status instances. This explication of state transitions can be used to answer competency question #3 (What number of Analyse Defect tasks is handled within two hours?).

```
?a[isOccurrenceOf hasValue AnalyseDefect] memberOf
ActivityOccurrence and ?b[isOccurrenceOf hasValue ?a] memberOf
OccurrenceStatus and ?c[isOccurrenceOf hasValue ?a] memberOf
OccurrenceStatus and ?b[hasStatus hasValue active] memberOf
OccurrenceStatus and ?c[hasStatus hasValue completed] memberOf
OccurrenceStatus and ?b[occursAt hasValue ?d] memberOf
OccurrenceStatus and ?c[occursAt hasValue ?e] memberOf
OccurrenceStatus and wsml#numericLessThan(?e,?d+7200).
```

Listing 4: Answering competency question #3

### 3.2 Behavioural dimension constituents

The behavioural dimension is concerned with the task ordering in a process model and the routing along those tasks. In XDPL as well as in the proposed ontologisation in oXPDL, the *Process* type constitutes the primary modelling element. It groups related activities, data, and resources together. *Activities* represent the reusable task behaviour in a process and take one of the following types, a triggered event, a block activity or a route activity. Route activities together with transitions between them constrain the ordering of activities in a process model.

However, the ordering in a workflow log is always totally ordered, since during execution only one possible path has been taken. In order to query the workflow log for behavioural properties, the ontological model needs to include explicit behavioural semantics. We adopted the axiom schemata from PSL [11] and relaxed its model semantics in accordance to our competency questions and the constraints imposed by the total ordering. The model includes the axioms of PSL defining a discrete ordering of timepoints. A transitive, irreflexive and totally ordered relation, *before*, is introduced for that purpose. The listing below shows the axioms related to the *before* relation.

$$\begin{aligned}
 & \textit{before}(T1, T3) : - \textit{before}(T1, T2) \textit{ and } \textit{before}(T2, T3) \textit{ and} \\
 & \textit{not } T1 := T2 \textit{ and } \textit{not } T1 := T3 \textit{ and } \textit{not } T2 := T3 \\
 & T1 := T2 : - \textit{before}(T1, T2) \textit{ and } \textit{before}(T2, T1) \\
 & ! - \textit{before}(T1, T1)
 \end{aligned}$$

Table 1: Axioms constraining *before* relation

An occurrence tree is a set of all discrete sequences of activity occurrences. The tree is constrained by the *earlier* and *initial* relation.

A *concrete* relation is introduced to constrain the tree to that occurrence ordering which actually occurred during execution. When a workflow log is converted to oXPDL+, concrete relations are added to the ontological model according to the execution order of the activities in the log based on their time occurrence. The set of concrete *activityOccurrences* constitute a single branch in the occurrence tree.

$earlier(S1, S3) : - earlier(S1, S2) \text{ and } earlier(S2, S3)$   
 $earlier(S1, S3) : - earlier(S1, S2) \text{ and } earlier(S3, S2) \text{ and}$   
 $not\ earlier(S3, S1) \text{ and } not\ S3 ::= S1$   
 $earlier(S3, S1) : - earlier(S1, S2) \text{ and } earlier(S3, S2) \text{ and}$   
 $not\ earlier(S1, S3) \text{ and } not\ S3 ::= S1$   
 $S3 ::= S1 : - earlier(S1, S2) \text{ and } earlier(S3, S2) \text{ and}$   
 $not\ earlier(S1, S3) \text{ and } not\ earlier(S3, S1)$

Table 2: Axioms constraining *earlier* relation

Based on that model we can query the model for the competency question #4 (Which tasks precede the “Archive Repair” in the instances of the “Repair” process) as follows.

```
?a[isOccurrenceOf hasValue Repair] memberOf ProcessOccurrence and
?a[hasActivityOccurrence hasValue ?b] memberOf ProcessOccurrence and
?a[hasActivityOccurrence hasValue ?c] memberOf ProcessOccurrence and
?b[isOccurrenceOf hasValue ArchiveRepair] memberOf
ActivityOccurrence and relationInstance before(?c,?b) and
relationInstance concrete(?c,?b).
```

Listing 5: Answering competency question #4

By inference on the axiom schemata, the reasoner interpreting the query returns all *ActivityOccurrences* preceding *ArchiveRepair*, also those which are not immediate predecessors and directly linked via a transition relation. Similarly we can query the model for competency question #5 (What is the most frequent path in the “Repair” processes?). It returns all possible occurrence paths. The statistical analysis would have to be done external to the reasoner.

```
?a[isOccurrenceOf hasValue Repair] memberOf ProcessOccurrence and
?a[hasActivityOccurrence hasValue ?b] memberOf ProcessOccurrence and
relationInstance before(?b,?b) and relationInstance concrete(?b,?b).
```

Listing 6: Answering competency question #5

Multiple choice paths in a workflow log can be observed by the fact that two or multiple activity occurrences switch the order over multiple workflow logs. The query in listing 7 returns such activity occurrences answering competency question #6.

```
?a[isOccurrenceOf hasValue Repair] memberOf ProcessOccurrence and
?a[hasActivityOccurrence hasValue ?b] memberOf ProcessOccurrence and
?a[hasActivityOccurrence hasValue ?c] memberOf ProcessOccurrence and
?b != ?c and relationInstance before(?b,?c) and relationInstance
concrete(?b,?c) and relationInstance before(?c,?b) and
relationInstance concrete(?c,?b).
```

Listing 7: Answering competency question #6

### 3.3 Resource dimension constituents

The resource dimension defines *who* is responsible for performing a task in a workflow and gives the modeler means to assign constraints on the agent (be it a human or a machine) responsible for carrying out a task. Resources are commonly grouped into roles and organisational units, whereas the workflow *participant* constitutes the focal modelling element in the resource dimension.

The organisational aspect in XPDL is only weakly defined; a process participant is simply a token. However, in the translation to oXPDL these types are modelled as independent classes which can be populated in oXPDL+. The most important entities to analyse in a workflow are human participants, who can be identified unambiguously in the ontological model by for example their FOAF profile<sup>3</sup>. Since this information is likely to be not provided in the workflow log, which commonly includes a person’s email address only, our implementation uses a semantic lookup service (i.e. Sindice.com [23]) to determine the FOAF URI of the person’s email. If this succeeds or if the information is added manually to the ontological model, a possible resource allocation to a workflow task could look as shown in listing 8.

```
instance SimpleRepair memberOf Activity
  hasName hasValue "Simple Repair"
  hasPerformer hasValue JuniorTechnician

instance JuniorTechnician memberOf Participant
  hasParticipantType hasValue Human

instance SimpleRepair041220071040 memberOf ActivityOccurrence
  hasPerformer hasValue JohnDoe

instance SimpleRepair051220070832 memberOf ActivityOccurrence
  hasPerformer hasValue JohnMurphy

instance JohnDoe memberOf Human
  hasProfile hasValue ."http://example.org/~johndoe/foaf.rdf#me"
```

Listing 8: Workflow log including human resources

Based on such a workflow log model competency question #7 (“Who works on the Simple Repair task in the Repair process”) can be answered by the query in listing 9, to retrieve all persons having performed the ‘Simple Repair’ task over multiple runs of the Repair process.

```
?a[isOccurrenceOf hasValue Repair191020070930] memberOf
ActivityOccurrence and ?a[hasPerformer hasValue ?b] memberOf
ActivityOccurrence and ?b[hasProfile hasValue ?c] memberOf Human.
```

Listing 9: Answering competency question #7

Resources can not only be humans, but also applications or if the process is analysed on a higher abstraction level, also organisational units. Query #8 retrieves the departments concerned with the “Analyse Defect” task, based on the affiliations of the involved process participants.

```
?a[isOccurrenceOf hasValue Repair] memberOf ActivityOccurrence and
?a[hasPerformer hasValue ?b] memberOf ActivityOccurrence and
?b[belongsTo hasValue ?c] memberOf Human.
```

Listing 10: Answering competency question #8

By querying multiple workflow logs as they are populated to the knowledge base (c.f section 5), one could for example retrieve all employees involved in the “Repair” process over multiple executions to answer competency question #9. The query is similar to the one shown in 9, but with the *isOccurrenceOf* relation ranging over the “Repair” process instead of a specific instance of it (i.e. Repair191020070930).

<sup>3</sup><http://www.foaf-project.org/>

## 4 Process Mining in oXPDL

In this section, we briefly describe two distinctive semantic process mining techniques. The first technique proposes to semantically annotate log attributes by binding them, as ontology instances, to their respective concepts in the business background ontologies. The second approach aims to construct process users' domain models (users' execution topology) by using ontologies created on the basis of user profile mining that describe a users's preferences and behaviour in a particular business domain.

### 4.1 Enriching oXPDL with Instance Recognition Semantics

This technique assumes that process logs are not semantically annotated; a realistic assumption considering that no current BPMS exports semantically annotated business processes. Our aim is to automatically recognize the business background ontology instances in logs. Indeed, semantic annotation research is fundamental for a semantic Business Process Management System (sBPMS). A semantic annotation step is needed to add formal metadata to logs which lack semantic description regarding the related business background. This metadata links data in log to the defined concepts in the business background ontologies. Thus reasoners will be capable of interpreting data in these semantically annotated logs with respect to their respective business background ontologies, as annotated content becomes machine-processable.

Although oXPDL+ provides a solid basis for BPMSs, it lacks sufficient declarative semantics for process execution instance specification and recognition. This omission prevents oXPDL+ from being a satisfactory ontology language for automated instance semantic annotation. We can resolve this problem by adding to ontological oXPDL+ declarations epistemological instance recognition semantics that include external representations and context recognition information for atomic, lexical ontology concepts from process logs. Our approach proposes an automated annotation using oXPDL+ ontologies for a rich instance-recognition semantics from non annotated process logs through Log Instance Semantics Recognisers (LISR).

LISR are formal specifications that identify instances of a concept in logs. The concept should be a lexical element of a formal ontology (e.g. concepts such as **date**, **time**, **place**, **performer**, **Electric Device**, **Analyse activity**, **Repair Department**, etc.). Thus, an LISR of an ontology concept (e.g. **Electric Device**) interprets an instance in a log fragment from Listing 2 (e.g. the device number in "hasName hasValue "Electronic Device 88" ") to have the intensional meaning of the defined concept (e.g. **Electric Device**). Listing 11 shows a partial ISR declaration for **Electric Device** concept. We use Perl-style regular expressions to declare recognition patterns.

```
Electric Device
external representation: [1-9]10
left context phrase: *E(lectr)?ic?|Dev(ice)?
right context phrase:\b
...
end
```

Listing 11: ISR Declarations for **Electric Device**

The use of instance recognition semantics of process ontologies over logs leads to enhanced knowledge sharing and reuse through the inter-organisation process logs. This Based-Instance Automated Annotation extension is epistemological in nature. While an ontology (in our context) focuses on defining formal specifications of facts and the relations between facts, epistemology (in our context) focuses on how we know a presented fact is about a formal specification, i.e.,

the recognition of instances. Ontological definitions are independent of the form of knowledge representation. In contrast, epistemological definitions are often sensitive to the form of knowledge representation [12].

## 4.2 Making users execution topology explicit

Collecting a users' interaction data during their process or activities instantiation and analysing it further is inevitable for understanding their behaviour and developing assistance systems. To reason about the collected information, techniques of process mining have to be applied. Hereby, we will concentrate on the usage and structure mining for process re-engineering and usage characterisation through users' profile mining.

How users are executing processes is heavily dependent on their own conceptual model of the subject area rather than the established process model. In other words, users have an implicit conceptual model of the domain in their mind. This model is based on their knowledge of the domain and does not entirely match to the given process model. There exists a well-known problem of possible mismatch between users domain model and initially designed process model. However, there is still lack of works devoted to making users domain models explicit. Research on semantic process is dealing with this issue by supporting process execution with domain ontologies, which should be commonly shared conceptualisations of a domain.

A possible solution to this problem is to provide a kind of personalisation service. Mining users' execution topology provides a solution to this mismatching problem by making process users' domain models (users' execution topology) explicit by using ontologies created on the basis of the user profile mining from logs. This technique enables to improve business ontologies and existing process models to become closer to the user preferences as well as derive new and specific ones.

We aim to enrich activity dependencies expressed in oXPDL+ semantically by annotating them by a users' execution frequencies. We claim that implicit domain models of process users, if made explicit, can be valuable for improving process models in order to bring them closer to the user preferences and making the process discovery efficient as well as helping to solve above-stated mismatch problem. Based on original log data mining procedure, user preferences can be found and concepts corresponding to a user profile ontology are automatically extracted from the mining result and added to the user profile ontology description. Using ontology reasoning services, extracted user profile concepts are classified under predefined user profile concepts. This allows specifying a rather general user profile concepts according to mined user preferences giving as a result a definition of users' execution topology.

This approach can be seen as a new method for learning users' execution topology in order to provide personalisation services. Comparing to existing semantic process, our approach makes users' execution topology (expressed in terms of ontologies) explicit instead of concentrating on static process domain ontologies explicit in the design phase. Novel ideas of our proposal lie in giving conceptual meaning to process usage mining results by using ontologies and automatic classification of concepts to ontologies via ontology reasoning as a part of the system.

## 5 Converter Implementation

In this section we first describe the conversion methodology and a simple architectural layout to retrieve both, the process model and one or many process instances from a process aware information system. Second, we briefly outline the conversion algorithm from XPDL to oXPDL to convert the

process model and from MXML to oXPDL+ to convert the process instances. The complete ontology specification and the translation tool can be found online<sup>4</sup>.

## 5.1 Methodology and Architecture

The analysis and mining techniques proposed in earlier sections require the process model and its instances to be available according to the oXPDL+ schema. Considering that current process aware systems do not output semantic Business Process Models, our conversion methodology assumes that the system at least supports export of its *proprietary* process model to XPDL. Since all process aware information systems log events in different formats, it is further required to translate the workflow logs to a uniform format. As pointed out earlier, XPDL lacks the conceptual elements to define workflow logs. Thus, none of the systems offering XPDL export functionality, can avail of a uniform format to export their run time events. The ProM framework [7] tackles this problem by offering 15 import filters to widely used process aware systems and WfMSs to import their internal workflow logs to MXML.

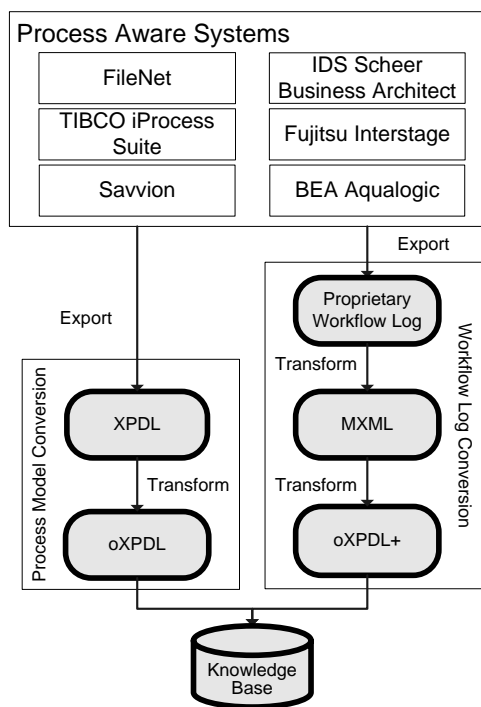


Figure 3: Conversion Methodology

Figure 3 describes the complete conversion process; first the proprietary process model is exported to an XML instance according to the XPDL schema. Second, the XPDL2WSML converter described in section 5.2 ensures a correct translation of this instance to its ontological representation according to oXPDL and exports the ontology to the knowledge base. During or after execution of the process model in the process aware system, the proprietary workflow log is converted to MXML, using the adapter framework provided in ProM. Finally, the MXML2WSML tool

<sup>4</sup><http://www.m3pe.org/oXPDL/>

described in section 5.3 converts the MXML instance to oXPDL+ and populates the knowledge base. The Workflow log conversion is repeated for every new execution of the process model.

### 5.2 Process Model conversion

The process model conversion involves two steps. Firstly, the oXPDL Core Ontology (CO) is loaded, serving as the language grammar including the concept hierarchy, its attributes and their cardinalities and secondly, the XPDL instance are analysed. The XML file is parsed to find children and sibling nodes. Every node is analysed, whether it is a *text node* or an *element node*. In both cases it is further checked if the concept or attribute is defined in CO. If it is, an attribute for the current instance or a new instance is created. The names of *element nodes* are mapped with a special dictionary including synonyms defined by the WfMC [21], to be able to map element names to the concepts in CO. After the name mapping the converter checks whether a concept is actually defined in CO or not. If yes, the process continues with the instance name construction. There are two cases that may occur. Either the ID is explicitly defined in the XML document or the tool generates a unique identifier by a combination of the element name and a hash of the creation date. When a new instance is created all attributes within the concept are parsed, their types are checked and if required they are added to the ontology.

The result of the conversion process is a knowledge base with the set of classes and properties from oXPDL and the associated instances created based on the XML input. The complete conversion algorithm of an XML instance according to the XPDL schema is illustrated in the activity diagram in figure 4.

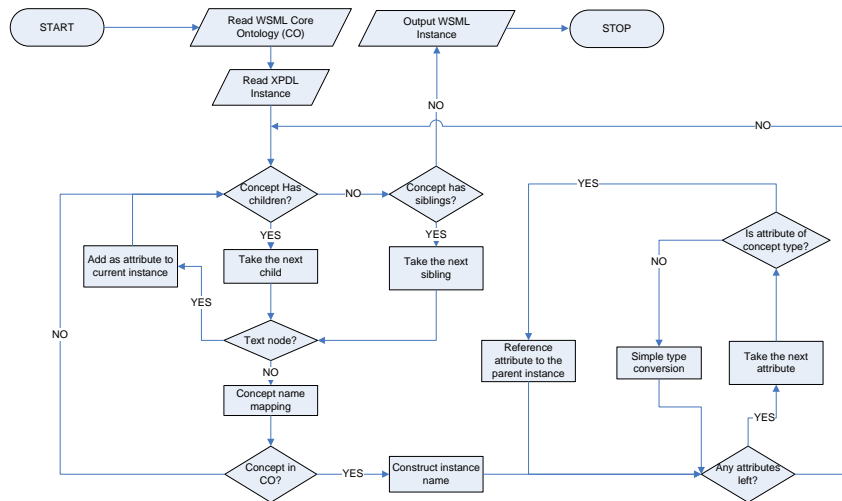


Figure 4: Processing steps of the conversion algorithm

### 5.3 Workflow Log conversion

As mentioned above we do not directly provide adapters to convert the logs of different process aware systems, but we take advantage of the adapters offered in ProM. This approach further benefits

the user, since he can not only avail of the analysis techniques shown in this paper, but also choose from an even bigger myriad of process mining techniques to analyse process instances already implemented in ProM. The MXML2WSML converter, however, ensures the correct translation from a log according to MXML to oXPDL+. The meta models are quite similar and the conversion is trivial. Figure 3 shows how the concepts in the two schemas relate.

MXML	oXPDL+
WorkflowLog	Package
Process	Process
ProcessInstance	ProcessOccurrence
WorkflowModelElement.activity	ActivityOccurrence
AuditTrailEntry.WorkflowModelElement	ActivityOccurrence.Name
AuditTrailEntry.EventType	OccurrenceStatus
AuditTrailEntry.Timestamp	OccurrenceStatus.Timepoint
AuditTrailEntry.Originator	ActivityOccurrence.Participant
Data.Attribute	DataFieldAccurrence
Data.Name	DataFieldAccurrence.Name

Table 3: The MXML mapping relations

## 6 Conclusion

Process analysis and process mining techniques aid in the (re-)design of processes in the business process modelling phase and in the (re-)configuration during the business process configuration phase. Since most WfMS have their own internal structure and language to define process models and workflow logs, when trying to use workflow logs from different systems, a standardised way to represent these workflow logs is needed. The lack of a standardised way to model workflow logs manifests itself since many enterprise information systems vendors increasingly focus on the analysis of process instances to allow what they call Business Activity Monitoring or Business Intelligence.

We have presented an approach to ontologise the XPDL standard, resulting in our oXPDL ontology to further extend the model with workflow log modelling capabilities in oXPDL+. We have explained the ontologisation methodology, based on a guiding set of “competence questions”, and have shown how oXPDL+ models can answer these questions on automatically translated XPDL process model and workflow log instances based on such models. As a result, oXPDL+ is a workflow model and process instance interchange model that explicitly captures intended workflow semantics, can be used with background ontologies, and can be analysed and mined for process-relevant competence questions. We presented a conversion algorithm from XPDL to oXPDL to convert the process model and from MXML to oXPDL+ to convert the process instances.

Currently, we are investigating a method for the semi-automatic construction of ontologies using logs of any business domain for the extraction of concepts and relations. By comparing the relative frequency of terms in logs with their typical, expected use, the method identifies concepts and relations and specifies the corresponding process ontology using oXPDL+ workflow logs.

## References

- [1] W. M. P. van der Aalst. The application of petri nets to workflow management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.

- [2] W. M. P. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, *et al.* Workflow mining: a survey of issues and approaches. *Data Knowl. Eng.*, 47(2):237–267, 2003.
- [3] A. Bonifati, F. Casati, U. Dayal, and M.-C. Shan. Warehousing workflow data: Challenges and opportunities. In *Proceedings of the 27th International Conference on Very Large Data Bases*, pp. 649–652. San Francisco, CA, USA, 2001.
- [4] D. Brickley and R. Guha, (eds.) *RDF Vocabulary Description Language 1.0: RDF Schema*. W3C Recommendation, 2004.
- [5] J. de Bruijn, H. Lausen, A. Polleres, and D. Fensel. The web service modeling language WSML: An overview. In *Proceedings of the European Semantic Web Conference (ESWC)*, pp. 590–604. 2006.
- [6] B. F. van Dongen and W. M. P. van der Aalst. A meta model for process mining data. In *Proceedings of the Open Interop Workshop on Enterprise Modelling and Ontologies for Interoperability*,. 2005.
- [7] B. F. van Dongen, A. K. A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, *et al.* The prom framework: A new era in process mining tool support. In *26th International Conference on Applications and Theory of Petri Nets (ICATPN 2005)*. Miami, FL, USA, 2005.
- [8] M. Dumas, W. M. P. van der Aalst, and A. H. ter Hofstede. *Process Aware Information Systems: Bridging People and Software Through Process Technology*. John Wiley & Sons, Hoboken, New Jersey, 2005.
- [9] J. Eder, G. E. Olivotto, and W. Gruber. A data warehouse for workflow logs. In *Proceedings of the First International Conference on Engineering and Deployment of Cooperative Information Systems*, pp. 1–15. 2002.
- [10] D. Grigori, F. Casati, M. Castellanos, U. Dayal, *et al.* Business process intelligence. *Computers in Industry*, 53(3):321–343, 2004.
- [11] M. Gruninger. Ontology of the process specification language. In S. Staab and R. Studer, (eds.) *Handbook on Ontologies*, pp. 575–592. Springer, 2004.
- [12] N. Guarino. Formal ontology, conceptual analysis and knowledge representation. *Int. J. Hum.-Comput. Stud.*, 43(5-6):625–640, 1995.
- [13] A. Haller, M. Marmolowski, E. Oren, and W. Gaaloul. oxpdl: a process model exchange ontology. Tech. Rep. DERI-TR-2007-12-12, DERI, 2007.
- [14] F. Leymann and D. Roller. *Production workflow: concepts and techniques*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2000.
- [15] D. L. McGuinness and F. van Harmelen, (eds.) *OWL Web Ontology Language*. W3C Recommendation, 2004.
- [16] A. K. A. de Medeiros, C. Pedrinaci, W. M. P. van der Aalst, J. Domingue, *et al.* An outlook on semantic business process mining and monitoring. In *OTM Workshops*, pp. 1244–1255. Vilamoura, Portugal, 2007.
- [17] M. Z. Muehlen. *Workflow-based Process Controlling: Foundation, Design and Application of workflow-driven Process Information Systems*. Logos, Berlin, 2004.

- [18] zur Muehlen Michael. Process-driven management information systems - combining data warehouses and workflow technology. In B. Gavish, (ed.) *Proceedings of the 4th International Conference on Electronic Commerce Research (ICECR-4)*, pp. 550–566. 2001.
- [19] A. P. Sheth, W. M. P. van der Aalst, and I. B. Arpinar. Processes driving the networked economy. *IEEE Concurrency*, 7(3):18–31, 1999.
- [20] S. Thatte *et al.* Business process execution language for web services. 2003.
- [21] The Workflow Management Coalition. Terminology and glossary. 1999.
- [22] The Workflow Management Coalition. Workflow Standard Process Definition Interface – XML Process Definition Language. 2005.
- [23] G. Tummarello, R. Delbru, and E. Oren. Sindice.com: Weaving the open linked data. In *6th International Semantic Web Conference*, pp. 552–565. Busan, South Korea, 2007.
- [24] M. Uschold and M. Grüninger. Ontologies: principles, methods, and applications. *Knowledge Engineering Review*, 11(2):93–155, 1996.