Integration of heterogeneous BPM Schemas: The Case of XPDL and BPEL

Thomas Hornung[1], Agnes Koschmider[2], and Jan Mendling[3]

[1] Institute of Computer Science, Albert-Ludwigs University Freiburg, Germany
hornungt@informatik.uni-freiburg.de
[2] Institute of Applied Informatics and Formal Description Methods
University of Karlsruhe (TH), Germany
koschmider@aifb.uni-karlsruhe.de
[3] Institute of Information Systems and New Media, WU Vienna, Austria
jan.mendling@wu-wien.ac.at

Abstract Heterogeneous Business Process Modeling (BPM) schemas have been a problem for business process management throughout the last couple of years. Although there are several standardization efforts in this area, none of the proposals is commonly accepted as a de facto standard in the industry. Methodological guidance is needed in order to consolidate concurrent schema proposals especially in the BPM ares. This paper discusses the applicability of schema integration for this purpose. We use the case of integrating XPDL 2.0 and BPEL 2.0 to highlight that schema integration is not able to cope with heterogeneous control flow representation of BPM schemas. As a consequence, we extend the schema integration process with a Schema Refactoring step that builds on the identification of transformation functions between schema constructs. This step leads to integrated BPM schemas with less constructs and that include only one control flow representation paradigm.

1 Introduction

Heterogeneity of schemas for business process modeling is a major problem for business process management [1] and triggered several academic efforts to compare (e.g. [2]) or to identify patterns (e.g. [3]) of business process modeling languages. Beyond that, standardization of business process modeling and workflow languages has been discussed for more than 10 years [4]. Various standardization bodies including WfMC[4], OASIS[5], UN/CEFACT[6], W3C[7], OMG[8], and BPMI[9] have proposed partially concurrent standards (for an overview see e.g. [2,5]).

[4] Workflow Management Coalition (WfMC): http://www.wfmc.org
[5] Organisation for the Advancement of Structured Information Standards (OASIS): http://www.oasis-open.org
[6] United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT): http://www.unece.org/cefact
[7] World Wide Web Consortium (W3C): http://www.w3.org
[8] Object Management Group (OMG): http://www.omg.org
[9] Business Process Management Initiative (BPMI): http://www.bpmi.org

Recently, there has been a trend towards consolidation of various standards. While BPEL [6] has combined concepts of Microsoft's XLANG [7] and IBM's WSFL [8], the new version BPEL 2.0 [9] also seems to consolidate concepts from BPML [10] whose author is among the BPEL 2.0 editors. Furthermore, the new XPDL Version 2.0 [11] extends the old version with concepts of BPMN [12] so that XPDL can be used as a serialization for BPMN diagrams. Finally, the Business Process Definition Metamodel [13] currently being defined by OMG aims to provide a UML stereotype for BPM that is mainly influenced by the BPMN and BPEL specifications.

Although this consolidation is desirable from an industry perspective, there has been criticism from a methodological, more academic point of view on the way how consolidation is achieved. In [14] the diverging strategies of different stakeholders in BPM standardization are highlighted with an emphasis on the bargaining character of such processes. In [15], XPDL in its version 1 is criticized for capturing only the minimal consensual set of control flow primitives. It would be more desirable to offer a superset of concepts found in practice as a standard. In the case of BPEL the control flow concepts of XLANG and WSFL were put together, but semantic redundancies were not eliminated. Accordingly, there is a choice in BPEL between a block structured and a graph structured specification of control flow [16]. These examples illustrate that methodological guidance is needed for the consolidation and integration of heterogeneous BPM metamodels. The non-triviality of this task has been illustrated in a previous contribution [17].

In this paper, we adopt ideas from schema integration and extend them for the specific requirements of BPM metamodel integration. Our contribution is twofold. First, we present an integration methodology that can be used to integrate and consolidate heterogeneous BPM metamodels. From a schema integration point of view, we identify conflicts of heterogeneous control flow representation which are specific to the intention of BPM metamodels. Second, we apply this methodology to the integration of XPDL 2.0 and BPEL 2.0. As these are the two major standards for process execution, we aim to contribute to a further consolidation in the area of BPM. Before this background the paper is structured as follows. In Section 2 we present the basic ideas of schema integration and the problems of integrating heterogeneous control flow representation. In Section 3 we provide an overview of XPDL 2.0 and BPEL 2.0. We use these two BPM standards to illustrate the BPM metamodel integration process reported in Section 4. In Section 5 we present the integrated metamodel derived from XPDL and BPEL and discuss it as a candidate standard. Section 6 relates the elaborations of this paper to other research. Eventually, Section 7 concludes the paper and gives an outlook on future research.

2 Problems of Schema Integration for BPM Metamodels

Schema integration refers to the construction of a global schema from a set of local schemas. In general, the local schemas are heterogeneous, i.e. semantically related concepts are captured by different local schemas in a different way, e.g.

using different names or different structure (cf. e.g. [18]). The global schema is expected to be *complete* in capturing all concepts of the local schemas, *minimal* by including semantically related concepts only once, and still *understandable* [19]. Discovering semantic relationships like equivalence, subsumption, intersection, disjointness, and incompatibility between concepts of local schemas plays a central role for schema integration process. In Section 1, we have outlined that BPEL is not minimal and it is not clear whether the new version of XPDL is complete.[10] Therefore, schema integration could be a promising methodology to arrive at an integrated BPM schema that is both complete and minimal.

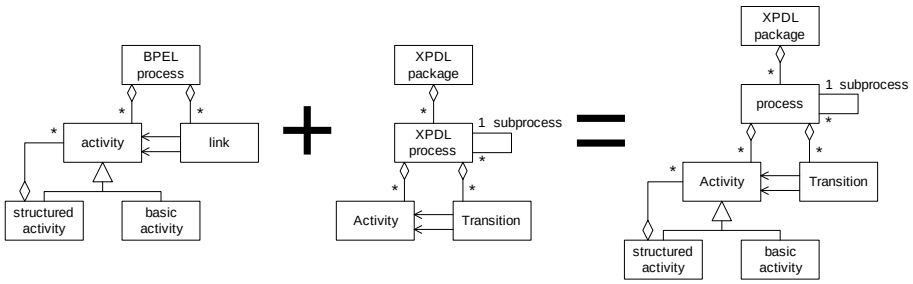

Figure 1. Parts BPEL and XPDL as well as integrated schema

Figure 1 shows a schema that could be constructed based on semantic relationships between the intentional domains [20] of XPDL and BPEL constructs.[11] There are three semantic relationships: the *BPEL basic activity* matches the *XPDL Activity*, the *BPEL link* matches the *XPDL Transition*, and the *BPEL process* matches the *XPDL process*. Yet, the integrated schema (right part of Figure 1) has still some deficiencies as further *simplifications* are possible. Some structured activities can be mapped to a set of XPDL activities and transitions; e.g. a *BPEL sequence* as one specific structured activity can be expressed as a sequence of XPDL activities connected by transitions. Therefore, the sequence is somehow redundant in the integrated schema. The problem is that this kind of redundancy cannot be expressed in terms of a binary semantic relationship, because a *BPEL sequence* has to be mapped to several *Activities* and *Transitions*. In order to eliminate this kind of redundancy, both XPDL and BPEL processes could be mapped to a language with more expressive modelling primitives like e.g. YAWL [21]. Another option could be a mapping to a more basic representation like state charts. Although this representational heterogeneity of behavioral aspects is typical for BPM languages, it seems that it is not inherent to behavior modelling only. Think of two car component schemas: one might use an unordered list of arcs and nodes (analogue to XPDL) to model subcom-

[10] XPDL 1.0 has been criticized for not being complete in [15].

[11] Please note that BPEL links are actually defined in flow structured activities. The schema is simplified for illustration purposes.

ponent relationships. The other uses a block-oriented representation by nesting components (analogue to BPEL). The latter allows to model a tree while the first accepts also general graphs. Intentional semantic relationships about mappings between these two different representations could help to eliminate redundant behavioral concepts in the integrated schema. This problems suggest that a straight-forward application of schema integration for static aspects does not yield the desired results.

3 Overview of BPEL and XPDL

BPEL is an executable language for the definition of a complex process as a composition from a set of Web Services.[12] BPEL Version 2 is currently being standardized by OASIS (see [9]). The main concepts of BPEL are *basic* and *structured activities*, *variables*, *partner links*, and *handlers*. Figure 2 illustrates these main concepts of BPEL and their interrelations by a UML class diagram.

In a simple case, a **BPEL process** defines partner links, variables, and activities. **Partner links** represent message exchange relationships between two parties. Via a reference to a partner link type the partner link defines the mutual required endpoints of a message exchange: the **myRole** and a **partnerRole** attributes defines who is playing which role. Partner links are referenced by basic activities that involve Web Service requests. **Variables** are used to store workflow data as well as input and output messages that are exchanged by Web Services activities via partner links. Furthermore, assign, throw, and rethrow activities can write to variables (not expressed in Figure 2. Scopes are specific **structured activities**[13] that can define local variables and handlers within their scope. **Handlers** specify responses to unexpected behavior like time or message events, faults, compensation, or termination. Nesting of structured activities is used to express control flow in BPEL. There are specific structured activities for loops (**while, forEach, repeatUntil**), sequential execution (**sequence**), conditional branching based on data (**if**) or events (**pick**), and concurrent branches (**flow**). Additional synchronization constraints in a flow can be defined via **links**. So-called **basic activities** specify the actual operations of a BPEL process. There are three activities involving Web Services: **invoke** for synchronous or asynchronous calls to a remote Web Service, **receive** to wait for the receipt of a specific message, and **reply** for responding to a remote request. All these activities reference a partner link and specify input and/or output variables for messages. The **correlation set** defines those parts of messages that identify the matching process instance. Handler activities can be used to **throw** or **rethrow** an fault which basically triggers a respective fault handler. The **compensate** activity triggers the compensation handler from within a fault handler. Furthermore, there are activities to **assign** data to variables, to **wait** for

[12] In this paper, we do not consider so-called abstract BPEL processes that can be used to model public aspects of business protocols.

[13] This specialization relationship is not expressed in Figure 2 in order not to blur the diagram.

a certain event before continuing, **empty** for doing nothing, **exit** to terminate the process, and **validate** for XML validation against a schema.

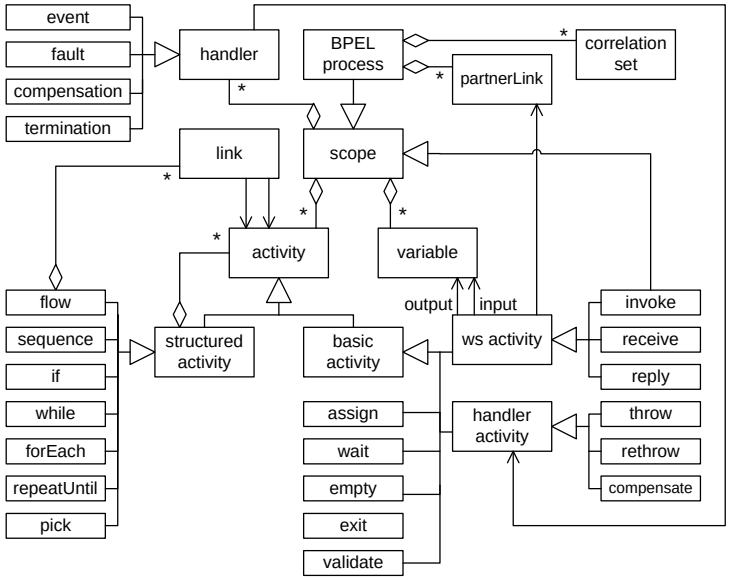

Figure 2. Metamodel of BPEL

XPDL was originally defined as an XML-based interchange format for Interface 1 of the Workflow Reference Model, specifying a "Minimum Meta Model" which "identifies commonly used entities within a process definition" [22]. In its new version 2.0 [11], XPDL has been enhanced to additionally serve as an interchange format for BPMN [12]. This realignment has been motivated by merger talks between the Workflow Management Coalition and the Business Process Management Initiative. As a consequence, the new XPDL version includes new concepts adapted from BPMN including e.g. Pools, Gateways, or Events. Figure 3 illustrates the main concepts of XPDL and their interrelations by a UML class diagram.

The XPDL **Package** serves as a container for all information associated with a process definition including Pools, Processes, Participants, Applications, Type Declarations, and Data Fields.[14] When used as a serialization of BPMN, the Package matches one Business Process Diagram (BPD). A **Process** (or Workflow Process) defines which Activities are executed and in which order. Since version 2, XPDL can include partner link elements similar to BPEL in a Process. The control flow is represented via **Transition** arcs between Activities. Transition conditions serve as guards for Transitions. The operations of an **Activity** can in-

[14] Associations, Artifacts, and PartnerLinkTypes are not represented in Figure 3.

volve Participants, Applications, and Data Fields which are associated with Type Declarations. XPDL distinguishes several types of Activities. The **Task/Tool** activity describes an activity that is executed automatically without humans being involved. The **Route** activity specifies join and split conditions. Similar to a Gateway, its sole purpose is to represent complex control flow conditions. It can specify both data- and event-based branching conditions. The **Block Activity** defines an embedded sub-process that executes an Activity Set. In contrast to that, the **Subflow** represents an explicit call to a sub-process that has been defined within the Package. Different types of **Events** are adapted from BPMN as special types of Activities. In contrast to BPEL, XPDL defines statistical information for Activities via the deadline, limit, and priority attribute that can be used by simulation engines. Furthermore, BPEL does not offer sub-processes. Yet, a respective extension has been proposed in the BPEL-SPE white paper [23]. Finally, XPDL includes explicit support for different application types like EJBs, Pojos[15], XSLT scripts while BPEL only considers Web Services.

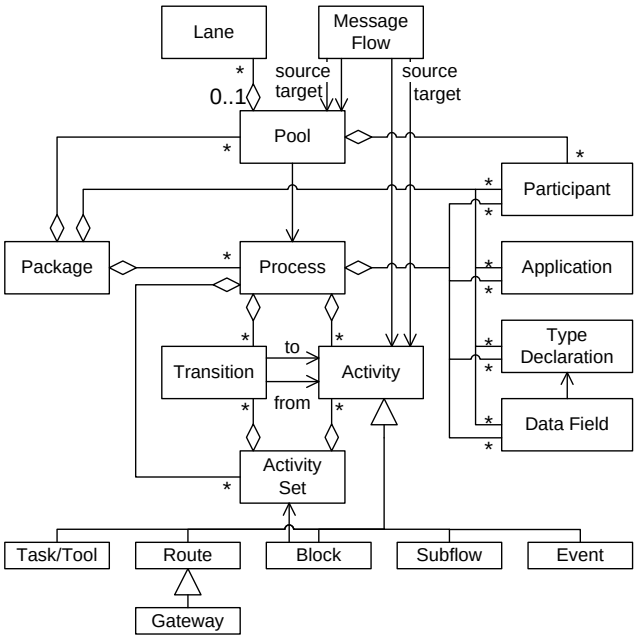

Figure 3. Metamodel of XPDL

[15] Plain old Java objects

4 BPM Metamodel Integration Process

In the previous section, we have seen that the integration of BPM metamodels has specific requirements: classical schema integration does not produce a global schema that is minimal with respect to control flow representation. In the following, we will essentially adopt and extend integration processes such as reported in [24,20,25]. As we are not interested in integrating the extension of a schema, we rely on semantic relationships defined on the intensional domains similar to [26,20]. Our BPM metamodel integration process includes the steps of (1) schema preparation, (2) schema matching, (3) schema merging, and (4) schema refactoring (see Figure 4).

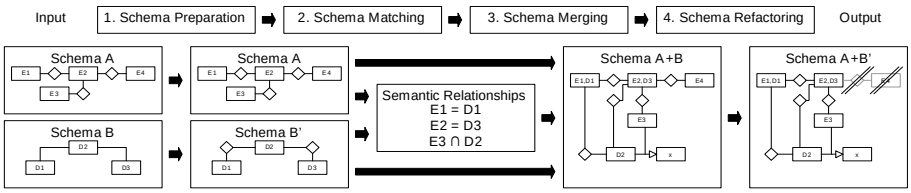

Figure 4. BPM Metamodel Integration Process

4.1 Schema Preparation

As a first step the two input schemas are transformed to a common data model. For our discussion of BPM metamodel integration, we map the BPM languages defined by an XML Schema to an object model using only a subset of elements offered by UML class diagrams. We use class diagrams basically because of UML's widespread adoption in the software engineering and information systems community. Therefore, it is well suited to communicate our integration process. Yet, the relational model or models especially tailored for integration like e.g. the generic integration model (GIM) [25] or the hypergraph data model (HDM) [20] could be used as well.

Figure 5 illustrates how BPEL links and XPDL transitions are represented in UML. The mapping from XML Schema to class diagrams eliminates heterogeneous XML representation of BPEL links and XPDL transitions. In BPEL the ends of links are associated with the source and target activities, in XPDL transitions have respective attributes to capture the identifying attributes of sources and targets. Furthermore, BPEL control flow can also be represented by structured activities like sequence. Schema Preparation as the mapping to a common data model has been automated in various research projects. Yet, XML leaves a lot of design choices. We preformed a manual transformation of the input schemas in order to assure that the semantics of the metamodels are captured properly.

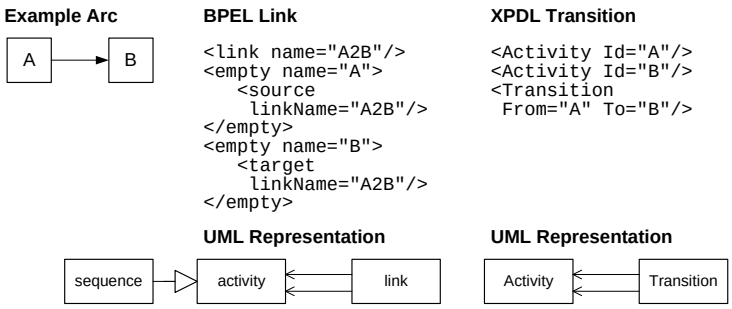

Figure 5. Schema Preparation of BPEL and XPDL control flow

4.2 Schema Matching

A central role for the derivation of the integrated schema plays the Schema Matching step. The two input schemas and the semantics of the schema concepts are compared in order to identify semantic relationships between the schema entities A and B. We consider semantic relationships that are defined on the intentional domains $D_i(A)$, i.e. the real world objects captured by the schema concepts. We adopt the definitions from [20].

- equivalence: two schema constructs A and B are equivalent, if and only if $D_i(A) = D_i(B)$. We write $A \overset{s}{=} B$.
- subsumption: schema construct A subsumes B, if and only if $D_i(B) \subset D_i(A)$. We write $B \overset{s}{\subset} A$.
- intersection: two schema constructs A and B are intersecting, if and only if $D_i(A) \cap D_i(B) \neq \emptyset, \exists C : D_i(A) \cap D_i(B) = D_i(C)$. We write $A \overset{s}{\cap} B$.
- disjointness: two schema constructs A and B are disjoint, if and only if $D_i(A) \cap D_i(B) = \emptyset, \exists C : D_i(A) \cup D_i(B) \subseteq D_i(C)$. We write $A \overset{s}{\not\cap} B$.

In the following we assume that the names of schema constructs are unique and only significant within their schema, i.e. the schema defines a namespace for schema constructs. This has the consequence that we do not need to define disjointness between homonymous elements in different schemas, but only equivalence, subsumption, and intersection. If you reconsider the graph-based control flow of XPDL and BPEL as given in Figure 5, there are four equivalences: $bpel{:}activity \overset{s}{=} xpdl{:}Activity$ and $bpel{:}link \overset{s}{=} xpdl{:}Transition$, as well as the two associations $bpel{:}source \overset{s}{=} xpdl{:}From$ and $bpel{:}target \overset{s}{=} xpdl{:}To$.

4.3 Schema Merging

This step takes the input schemas and merges them according to the semantic relationships identified in the Schema Matching step. We adopt the generic schema merging rules formalized in [20]. For further information on schema optimization by schema restructuring rules we refer to [20]. We do not consider restructuring here.

– equivalence: if $A \overset{s}{=} B$ then merge A and B to one construct in the integrated schema including all relationships of A and B.
– subsumption: if $A \overset{s}{\subset} B$, then include A and B in the integrated schema with a subclass relationship between B and A.
– intersection: if $A \overset{s}{\cap} B$, then include A and B in the integrated schema and add a new construct C to represent the common intentional domain with C being a superclass of both A and B.
– disjointness: if $A \overset{s}{\not\cap} B$, then include A and B in the integrated schema and add a new construct C that is a superclass of both A and B.

4.4 Schema Refactoring

Applying the schema merging rules results in the integrated model as defined in Figure 6. The problem of this model is that there are still redundancies is control flow representation that cannot be expressed as equivalence, subsumption, intersection, or disjointness semantic relationships. We address this problem by introducing a transformation function $t : \mathbb{P}(S) \rightarrow S$ such that S denotes the set of all constructs of the integrated schema. We say that for $R \subset S$ there is a transformation $t(R) = T$ with $T \in S$ if and only if the intentional semantics of T can be represented by the constructs included in R. In the example, the intentional semantics of a *BPEL sequence* can be expressed by a set of control flow arcs (*link* or *Transition*). We write $t(link) = sequence$. Each schema construct T that can be represented by other schema constructs R, i.e. if $t(R) = T$ exists, we exclude T from the integrated schema. This implies that the sequence structured activity would not be included in the final schema.

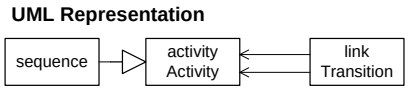

Figure 6. Result of Schema Merging of BPEL and XPDL control flow

5 Integrated Metamodel of XPDL and BPEL

In the previous section, we have presented a BPM integration process including the four steps of Schema Preparation, Schema Matching, Schema Merging, and Schema Refactoring. In this section, we present the results of applying this process to XPDL and BPEL. The results of manual Schema Preparation (*step 1*) is already given in Figures 2 and 3 of Section 3.

The next step of Schema Matching (*step 2*) is dedicated to the identification of semantic relationships between constructs of XPDL and BPEL. Table 1 illustrates some of these semantic relationships. The BPEL constructs **rethrow**,

validate and XPDL constructs **Artifact**, **Lane**, **Pool** and **SubFlow** cannot be matched with any construct of the other schema. The main equivalence relationships are $bpel.activity \stackrel{s}{=} xpdl.Activity$, $bpel.scope \stackrel{s}{=} xpdl.ActivitySet$, $bpel.link \stackrel{s}{=} xpdl.Transition$, $bpel.variable \stackrel{s}{=} xpdl.dataField$, and $bpel.process \stackrel{s}{=} xpdl.Process$.

Table 1: Schema Matching of BPEL and XPDL elements

XPDL [11]	**B**PEL [9]	Semantic Relationship
Activity	activity	$X \stackrel{s}{=} B$
ActivitySet	scope	$X \stackrel{s}{=} B$
Annotation	documentation	$X \stackrel{s}{=} B$
Application	WSDL.operation	$B \stackrel{s}{\subset} X$
Assignment	assign	$X \stackrel{s}{=} B$
DataField	variable	$X \stackrel{s}{=} B$
Event	$\{catch, catchAll,$ $compensate,$ $compensationHandler$ $eventHandlers, exit,$ $faultHandlers, onEvent,$ $onAlarm,$ $terminationHandler,$ $throw, wait\}$	$B \stackrel{s}{\subset} X$
Loop	$\{repeatUntil, while\}$	$B \stackrel{s}{\subset} X$
MessageFlow	correlationSet	$X \stackrel{s}{=} B$
Participant	partnerLink	$B \stackrel{s}{\subset} X$
Package	process	$B \stackrel{s}{\subset} X$
PartnerLink	partnerLink	$X \stackrel{s}{=} B$
PartnerLinkType	partnerLinkType	$X \stackrel{s}{=} B$
Process	process	$X \stackrel{s}{=} B$
Route	$\{empty, flow, if, pick\}$	$B \stackrel{s}{\subset} X$
Task	$\{invoke, receive, reply\}$	$B \stackrel{s}{\subset} X$
Transition	link	$X \stackrel{s}{=} B$
TypeDeclaration	XML.SchemaType	$X \stackrel{s}{=} B$

Based on the identified semantic relationships the XPDL and BPEL schema are merged (*step 3*). The result is given in Figure 7 and reflects the output of a classical schema integration approach. As already outlined in Section 2, the integrated schema is quite large and still includes redundant constructs. In the final Schema refactoring step (*step 4*) such redundancies are identified and removed. We have identified the following transformation functions:

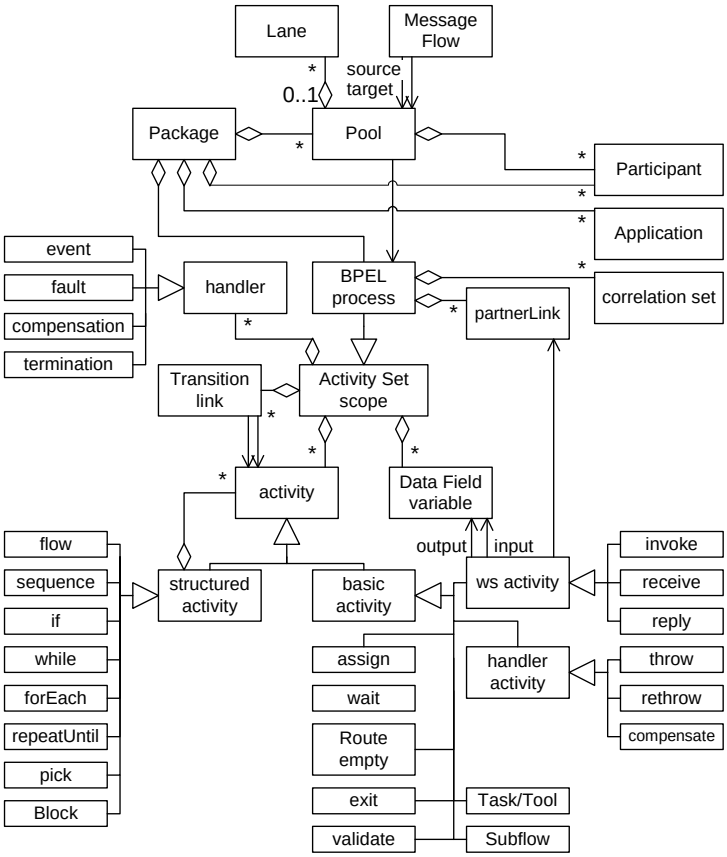

Figure 7. Merged Schemas of BPEL and XPDL

– All structured activities can be expressed by control flow arcs (*Transition/link*). That implies that $t(\{link, route\}) = structured\ activity$. Accordingly, structured activities can be excluded from the integrated schema.
– Activity Sets/scopes are special kinds of embedded processes. Their behavior can be modelled by independent sub-processes. That implies $t(process) = scope$. Thus, scopes can be dropped from the integrated schema.

These two Schema Refactoring operations yield a much simpler schema with less schema constructs. Control flow is represented in a graph-oriented way using *Transitions/links*. Furthermore, handlers can only be defined for processes while in the input BPEL schema, they could be defined for the process, the scope, and also for invoke activities.

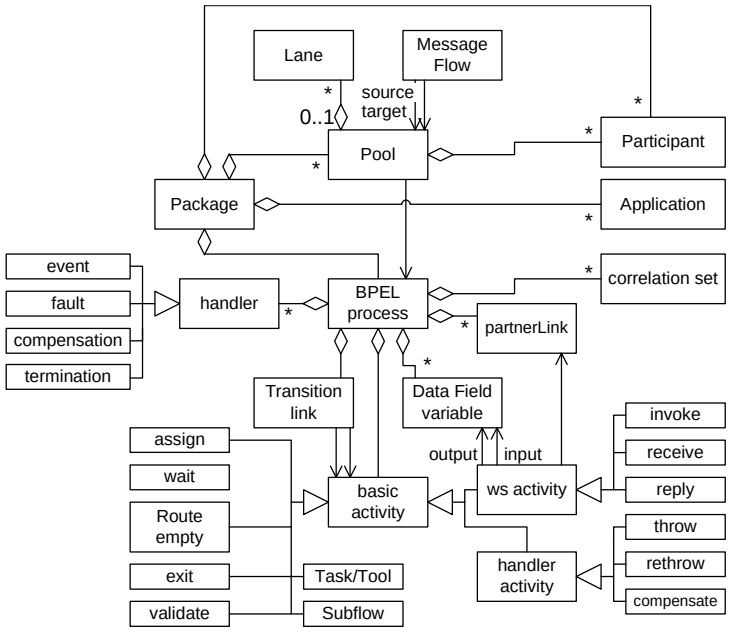

Figure 8. Refactored integrated Schema of BPEL and XPDL

6 Related Work

Numerous approaches dealing with the integration of heterogenous schemas have been published so far. Different data models for schema integration have been proposed, e.g. GIM [25] or HDM [20]. We refer to [27] for a detailed overview of different strategies in the context of (semi-)automatic schema matching. A good overview of research on schema integration in general can be found in [25].

The integration of behavioral aspects has received less attention in comparison to integration of static data models. Preuner et al. [28] presents an integration strategy for business process models given as a Petri net derivative called object/behavior diagrams (OBD). Yet, heterogeneity of business process modelling schemas is not discussed in this context. Integration is often related to some notion of inheritance. In [29] four types of inheritance relationships are defined for Petri nets. This work is motivated by model checking as it does not discuss integration aspects. In contrast to that Simon [30] presents an integration methodology for Module nets, a specific Petri net variant. Again, this work addresses the integration of models, but not the integration of metamodels or schemas for BPM. In so far, our paper complements this stream of work.

In the context of heterogeneous BPM schemas, a lot of research is dedicated not to integration directly, but to semi-formal comparisons. Examples include comparisons of XPDL, BPEL, and BPML in [31] and of several BPM languages in [2]. Another approach is taken by [3] who identify workflow patterns for control flow semantics. These patterns have been applied as a framework for comparing various BPM languages. Furthermore, that research inspired the specification of a new workflow language called YAWL that is able to capture all pattern (excluding implicit termination). Beyond that, there has been some work on transformations. An overview of transformations between BPEL and graph-oriented BPM languages can be found in [32].

Yet, there is doubt whether schema integration is suitable as a methodology for standardization of schemas. In [33] schema integration as a bottom-up methodology is contrasted with top-down domain modelling. Schema integration is said to produce schemas that are too much influenced by the local schemas and therefore rather difficult to understand, while domain modelling yields much clearer schemas. We explicitly pick up this criticism by introducing the fourth step of Schema Refactoring in our integration process. It combines the advantages of both schema integration and domain modelling.

7 Conclusion and Future Work

In this paper we have presented a BPM schema integration process that is able to cope with heterogeneous control flow representation of BPM schemas. The process extends the work on conceptual model transformations as presented in [20]. We introduce a novel step for Schema Refactoring that is guided by transformation functions between redundant schema constructs. The steps include (1) *Schema Preparation*, i.e. transformation the two input schemas to a common data model, e.g. UML, (2) *Schema Matching*, i.e. the identification of semantic relationships including equivalence, subsumption, intersection and disjointness between elements of the two schemas, (3) *Schema Merging*, i.e. the derivation of an integrated schema based on the semantic relationships, and (4) *Schema Refactoring*, i.e. the identification of transformation functions between schema constructs and the removal of these unnecessary constructs.

We have demonstrated the applicability of the process by integrating XPDL 2.0 and BPEL 2.0, two major competing BPEL schemas proposed by the Workflow Management Coalition and OASIS. The Refactoring step leads to a much simpler schema with less constructs that classical schema integration would yield. Still, no semantics are lost. Right now we are looking into other BPM standard candidates and try to incorporate them into the integrated model given in Section 5 in order to define a generic upper-bound BPM schema. The final goal of our work is to implement a workflow engine that can execute this generic language and that offers transformations to different standards.

References

1. Delphi Group: BPM 2003 – Market Milestone Report. White Paper (2003)
2. Mendling, J., Nüttgens, M., Neumann, G.: A Comparison of XML Interchange Formats for Business Process Modelling. In Feltz, F., Oberweis, A., Otjacques, B., eds.: Proceedings of EMISA 2004 - Information Systems in E-Business and E-Government. Volume 56 of Lecture Notes in Informatics. (2004)
3. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow Patterns. Distributed and Parallel Databases **14** (2003) 5–51
4. Hollingsworth, D.: The Workflow Reference Model: 10 Years On. In: The Workflow Handbook 2004. Workflow Management Coalition (2004) 295–312
5. Mendling, J., zur Muehlen, M., Price, A.: Standards for Workflow Definition and Execution. In: Process Aware Information Systems: Bridging People and Software Through Process Technology. Wiley Publishing (2005)
6. Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weerawarana, S.: Business Process Execution Language for Web Services, Version 1.1. Specification, BEA Systems, IBM Corp., Microsoft Corp., SAP AG, Siebel Systems (2003)
7. Thatte, S.: XLANG: Web Services for Business Process Design. Spec., Microsoft (2001)
8. Leymann, F.: Web Services Flow Language (WSFL). Spec., IBM Corp. (2001)
9. Arkin, A., Askary, S., Bloch, B., Curbera, F., Goland, Y., Kartha, N., Liu, C.K., Thatte, S., Yendluri, P., Yiu, A.: Web services business process execution language version 2.0. wsbpel-specification-draft-01, OASIS (2005)
10. Arkin, A.: Business Process Modeling Language (BPML). Spec., BPMI (2002)
11. Workflow Management Coalition: Workflow Process Definition Interface – XML Process Definition Language. Document Number WFMC-TC-1025, October 3, 2005, Version 2.00, Workflow Management Coalition (2005)
12. White, S.A.: Business Process Modeling Notation. Specification, BPMI.org (2004)
13. Koethe, M.R.: Business Process Definition Metamodel. Request for Proposals (bei/2003-01-06), Object Management Group (2003)
14. zur Muehlen, M., Nickerson, J.V., Swenson, K.D.: Developing Web Services Choreography Standards - The Case of REST vs. SOAP. Decision Support Systems (2005)
15. van der Aalst, W.M.P.: Patterns and XPDL: A Critical Evaluation of the XML Process Definition Language. QUT Technical report FIT-TR-2003-06, Queensland University of Technology, Brisbane (2003)
16. van der Aalst, W.M.P.: Don't go with the flow: Web services composition standards exposed. IEEE Intelligent Systems **18** (2003) 72–76

17. Mendling, J., Pérez de Laborda, C., Zdun, U.: Towards an Integrated BPM Schema: Control Flow Heterogeneity of PNML and BPEL4WS. In Althoff, K.D., Dengel, A., Bergmann, R., Nick, M., Roth-Berghofer, T., eds.: Post-Proceedings of the 3rd Conference Professional Knowledge Management (WM 2005). Volume 3782 of Lecture Notes in Artificial Intelligence., Springer Verlag (2005) 570–579
18. Kim, W., Seo, J.: Classifying schematic and data heterogeneity in multidatabase systems. IEEE Computer **24** (1991) 12–18
19. Batini, C., Lenzerini, M., Navathe, S.B.: A Comparative Analysis of Methodologies for Database Schema Integration. ACM Computing Surveys **18** (1986) 323–364
20. Rizopoulos, N., McBrien, P.: A general approach to the generation of conceptual model transformations. In Pastor, O., e Cunha, J.F., eds.: Advanced Information Systems Engineering, 17th International Conference, CAiSE 2005, Porto, Portugal, June 13-17, 2005, Proceedings. Volume 3520 of Lecture Notes in Computer Science., Springer (2005) 326–341
21. van der Aalst, W.M.P., ter Hofstede, A.H.M.: YAWL: Yet Another Workflow Language. Information Systems **30** (2005) 245–275
22. Workflow Management Coalition: Workflow Process Definition Interface – XML Process Definition Language. Document Number WFMC-TC-1025, October 25, 2002, Version 1.0, Workflow Management Coalition (2002)
23. Kloppmann, M., Koenig, D., Leymann, F., Pfau, G., Rickayzen, A., von Riegen, C., Schmidt, P., Trickovic, I.: WS-BPEL Extension for Sub-processes BPEL-SPE. Joint white paper, IBM and SAP (2005)
24. Sheth, A.P., Larson, J.A.: Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Comput. Surv. **22** (1990) 183–236
25. Schmitt, I., Saake, G.: A comprehensive database schema integration method based on the theory of formal concepts. Acta Inf. **41** (2005) 475–524
26. Larson, J.A., Navathe, S.B., Elmasri, R.: A theory of attribute equivalence in databases with application to schema integration. IEEE Trans. Software Eng. **15** (1989) 449–463
27. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB Journal **10** (2001) 334–350
28. Preuner, G., Conrad, S., Schrefl, M.: View integration of behavior in object-oriented databases. Data Knowl. Eng. **36** (2001) 153–183
29. van der Aalst, W.M.P.: Inheritance of business processes: A journey visiting four notorious problems. In Ehrig, H., Reisig, W., Rozenberg, G., Weber, H., eds.: Petri Net Technology for Communication-Based Systems - Advances in Petri Nets. Volume 2472 of Lecture Notes in Computer Science., Springer (2003) 383–408
30. Simon, C.: Incremental Development of Business Process Models. In Desel, J., Frank, U., eds.: Proceedings of the Workshop Enterprise Modelling and Information Systems Architectures. Volume 75 of Lecture Notes in Informatics., Klagenfurt, Austria, German Informatics Society (2005) 222–235
31. Shapiro, R.: A Comparison of XPDL, BPML and BPEL4WS. Draft version 1.4, Cape Visions, http://xml.coverpages.org/Shapiro-XPDL.pdf (2002)
32. Mendling, J., Lassen, K., Zdun, U.: Transformation strategies between block-oriented and graph-oriented process modelling languages. Technical Report JM-2005-10-10, WU Vienna (2005)
33. Hasselbring, W.: The role of standards for interoperating information systems. In Jakobs, K., ed.: Information Technology Standards and Standardization: A Global Perspective. Idea Group Publishing, Hershey, PA (2000) 116–130