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. Methodological guidance is needed in order to consolidate concurrent schema proposals especially in the BPM area. 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. We introduce a schema refactoring step that leads to integrated BPM schemas with less constructs.

1 Introduction

Heterogeneity of schemas for business process modeling (BPM) is a major problem for business process management [1] and triggered several academic efforts to compare BPM languages (e.g. [2]) or to identify patterns (e.g. [3]). There is doubt whether standards proposed by industry consortia can provide the required integration of BPM languages; at least the way this consolidation is achieved is questionable from a methodological, more academic point of view (see e.g. [4]). Therefore we propose a more theoretical approach to the integration of BPM metamodels which is based on schema integration and includes extensions to address the specifics of BPM languages. Our contribution in this context is twofold. First, we present an integration methodology that can be used to integrate heterogeneous BPM metamodels (Section 2). 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 [5] and BPEL 2.0 [6] and sketch the result in Section 3. For details refer to the long version of this paper[4]. Section 4 concludes the paper and gives an outlook on future research.

[4] http://wi.wu-wien.ac.at/˜mendling/publications/TR-Caise06.pdf

2 BPM Metamodel Integration Process

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. [7]). 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 *understandable* [8]. In the following, we adopt and extend integration processes such as reported in [9], relying on semantic relationships defined on the intensional domains similar to [10]. Our BPM metamodel integration process includes (1) schema preparation, (2) schema matching, (3) schema merging, and (4) schema refactoring.

2.1 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. Yet, the relational model or models especially tailored for integration such as HDM [10] could be used as well.

2.2 Schema Matching

In the schema matching step the two input schemas and the semantics of the schema constructs are compared in order to identify semantic relationships. In the following A refers to a construct of the first and B of the second schema. We consider semantic relationships that are defined on the intentional domains $D_i(A)$, i.e. the real world objects captured by the schema constructs. We adopt the definitions from [10]. As we assume names to be unique and significant only within each schema, we do not address homonyms. Therefore, disjointness only covers two non-overlapping constructs sharing a common super construct.

- 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 $A \overset{s}{\not\cap} B$.

2.3 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 [10] without considering 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.

2.4 Schema Refactoring

Applying the schema merging rules results in the integrated model as defined in Figure 1. 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 $t : \mathbb{P}(S) \to S$ 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.

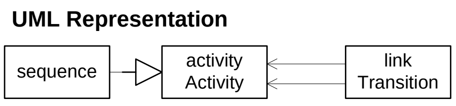

Figure 1. Result of schema merging of BPEL and XPDL control flow

3 Integrated Metamodel of BPEL 2.0 and XPDL 2.0

Figure 2 shows a part of the refactored integrated metamodel of BPEL and XPDL. We have identified among others the following transformation function: 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. The schema refactoring operations yield a much simpler schema with less schema constructs. Control flow is represented in a graph-oriented way using *Transitions/links.* This improves the integrated metamodel in terms of minimality and understandability.

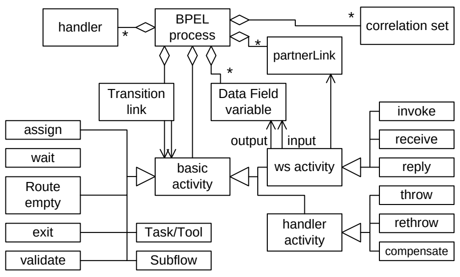

Figure 2. Part of the refactored integrated metamodel BPEL and XPDL

4 Conclusion and Future Work

In this paper we have presented a BPM metamodel 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 [10]. We have introduced a novel step for schema refactoring that is guided by transformation functions between redundant schema constructs. We have demonstrated the applicability of the process by integrating XPDL 2.0 and BPEL 2.0. The refactoring step leads to a much simpler schema with less constructs that classical schema integration would yield. In future research, we aim to provide tool support for the BPM metamodel integration process defined in this paper.

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.: EMISA 2004, Proceedings. LNI 56 (2004) 129–140
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. zur Muehlen, M., Nickerson, J.V., Swenson, K.D.: Developing Web Services Choreography Standards - The Case of REST vs. SOAP. Dec. Sup. Sys. **40** (2005) 9–29
5. 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)
6. Workflow Management Coalition: Workflow Process Definition Interface – XML Process Definition Language. WFMC-TC-1025, Oct. 3, Version 2.00, WfMC (2005)
7. Kim, W., Seo, J.: Classifying schematic and data heterogeneity in multidatabase systems. IEEE Computer **24** (1991) 12–18
8. Batini, C., Lenzerini, M., Navathe, S.B.: A Comparative Analysis of Methodologies for Database Schema Integration. ACM Computing Surveys **18** (1986) 323–364
9. Sheth, A.P., Larson, J.A.: Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Comp. Surv. **22** (1990) 183–236
10. Rizopoulos, N., McBrien, P.: A general approach to the generation of conceptual model transformations. In Pastor, O., e Cunha, J.F., eds.: CAiSE 2005, Proceedings. LNCS 3520, Springer (2005) 326–341