Integrating Privacy Policies into Business Processes

Michele Chinosi and Alberto Trombetta

Università degli Studi dell'Insubria
Dipartimento di Informatica e Comunicazione
via Mazzini 5, 21100 Varese, Italy

Abstract. The increased interest around business processes management and modeling techniques has brought many organizations to make significant investments in business process modeling projects. One of the most recent proposal for a new business process modeling technique is the *Business Process Modeling Notation* (BPMN). Often, the modeled business processes involve sensible information whose disclosure is usually regulated by privacy policies. As such, the interaction between business processes and privacy policies is a critical issue worth to be investigated. Towards this end, we introduce a data model for BPMN and a corresponding XML-based representation (called *BPeX*) which we use to check whether a BPeX-represented business process is compliant with a P3P privacy policy. Our checking procedures are very efficient and require standard XML technology, such as XPath.

1 Introduction

The ever-increasing interest around business processes management and modeling techniques has brought many organizations to make significant investments in business process modeling efforts. The *Business Process Management* (BPM) has been identified as one of the most important business priorities. The Workflow Management Coalition (WfMC) defines BPM as *a set of one or more linked procedures or activities which collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships* [1]. As such, it introduces methods, tools and techniques to support the development and the analysis of operational business processes. Inside this context, business process modeling techniques and languages are of absolute relevance. Again, WfMC defines business process modeling as *the time period when manual and/or automated (workflow) descriptions of a process are defined and/or modified electronically* [1]. One of the most recent proposal for a business process modeling technique is Business Process Modeling Notation (BPMN), adopted as standard by OMG [2].

The adoption of BPMN as a standard allows companies to define complex business processes possibly encompassing different administrative boundaries and requesting non-public data whose access is regulated by security-driven policies. In particular, privacy-related user data represent a relevant asset for companies and administrations. As such, in the recent years, several efforts have been made in order to provide mechanisms for expressing (and in some cases, enforcing) privacy policies protecting such

Chinosi M. and Trombetta A. (2008).
Integrating Privacy Policies into Business Processes.
In *Proceedings of the 6th International Workshop on Security in Information Systems*, pages 13-25
DOI: 10.5220/0001743900130025

data. One of the most well-known efforts in such direction is the P3P privacy policy description language [3]. P3P permits to represent privacy policies in an XML tree, basing on an XML-Schema model, which can be published by service providers to give users the capability to check automatically if the policies accomplish with the user preferences.

Therefore, it becomes a relevant, non-trivial issue to check whether a given complex business process (describing how processes interact and what data items they access) is compliant with a stated privacy policy (describing what processes are entitled to access which data items, provided that the corresponding purposes and obligations have been stated).

In this work we address the above mentioned issue by presenting a single framework in which both a business process and a corresponding privacy policy can be expressed and the compliance of the former with respect to the latter can be checked. We accomplish this by expressing both business processes and privacy policies in suitable XML formats and then proceed to check their compliance. We assume that privacy policies are expressed in P3P format. Regarding business processes, we do not rely on already disposable XML-based representations of BPMN, such as BPEL4WS [4] or XPDL [5, 6]. Both of them have disadvantages: BPEL4WS is strictly less expressive than BPMN, since only one single business process can be represented, and only one subset of BPMN elements can be deployed [7–9]. XPDL supports a larger fragment of BPMN but it does not render properly the hierarchical logical relationships between elements as well as it is not an executable language [5, 6]. Thus, XPDL is not the best way to represent BPDs if the goals are analysis, execution, extensions.

Hence, we present a new XML-oriented model, called *BPeX*, that faithfully describes *all* the relevant features of BPMN, such as the complete mapping of all the elements provided by the specifications and a tree model which reflects the elements dependencies and the hierarchical structure of diagrams. These features introduce some useful capabilities such as the possibility to export and share the diagram with other tools and to investigate processes to pinpoint bottle-neck or dead-locks (and consequently to patch them).

Having presented in a single, coherent framework both business processes and privacy policies, we then define the procedures for checking the compliance of a BPeX-based business process with respect to a P3P-based privacy policy. Such procedures, relying on the unified XML-based format employed for business processes and privacy policies are implemented using standard XML query languages, such as XPath.

2 Related Works

This paper starts from the official BPMN specifications as approved by OMG in 2006 and builds upon it. We don't assume any formal semantics underlying BPMN [10–12], rather we consider the semantics as it is presented in natural language in the OMG documentation. We give for some elements a more formal definition in order to perform some kind of queries and operations.

Other works about integrating privacy policies with business processes have been published since 2002 [13–15] even though the first work proposing an algorithm to

verify P3P policies on BPEL4WS tree was published in 2006 [16]. Moreover, in [17] an approach to extract RBAC models from BPEL4WS processes for the role engineering process is presented.

Due to space limitations, you can find more detailed information about P3P and BPMN on respective standard web-pages, [3] and [2]. For the same reason, we omit in this work almost all the code and the pictures explaining our model. It is possible to find them in the corresponding project web site under SourceForge repository at `http://bpex.sourceforge.net`. We will refer in this paper to the content published there with [18]. On the same web-site you can also find an extended version of this paper.

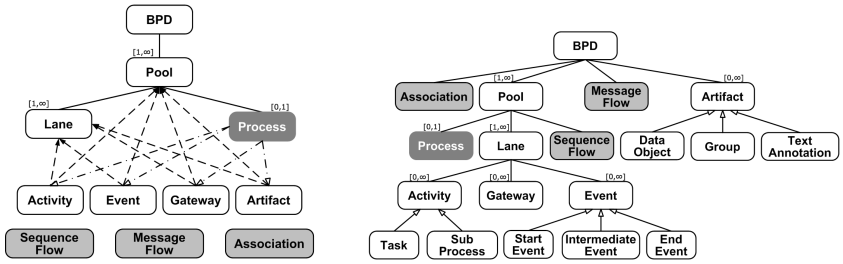
3 BPeX: a BPMN XML Linearization

At the present moment, BPMN defines simply a graphical notation without an explicit definition of the underlying meta model; that is, what are the basic elements composing a business process and what are the relationships among them. Clearly, providing a BPMN model is a first, necessary step in order to precisely state what is the meaning (or, more precisely, behavior) of a business process described as a BPMN diagram. As such, it is an interesting problem in itself to define such a suitable, BP-oriented model [10–12]. There are some research efforts aiming at the definition of a comprehensive model representing *all* the main features of BPM, but all such efforts build upon existing incomplete and/or inadequate formats. We have chosen a clean start by looking closely into BPMN and building our model from scratch, thus obtaining a clear model natively supporting all the relevant features of BPMN.

The result of our efforts is called *BPeX* and it is defined in a top-down fashion. We start pointing out all the different BPMN symbol families, we then proceed refining them through the definition of more precise symbol families, connected in a suitably defined hierarchy. Then, we add a representation of flows, adopting the same methodology.

We provide an XML version of such a model, in order to obtain a complete schema representing all the BPMN elements. The chosen hierarchical structure among BPMN elements (and flows) can be represented in a very natural way using XML-Schema. With a slight abuse of notation we call BPeX both the model and its XML-based version. In the following sections we will mention the BPeX features relevant for the present framework, namely how to smoothly integrate in it information about privacy policies. Any further information can be found at the BPeX project site [18].

Figure 1 shows a comparison between BPMN (as understood from the available documentation) on Fig. 1(a) and our BPeX meta model on Fig. 1(b). As it is possible to see, in BPeX model all the elements are connected to each other, while in the BPMN specifications most of the elements are only referenced by an external numerical value (represented graphically with dashed arrows) and, thus, they do not accomplish to represent the diagram structure. One of the most immediate advantages on using a full hierarchical model is, e.g., the complexity on check what Lane an element (an Event, an Activity or a Gateway) belongs to. Using BPMN model, one has to control the values of the Pool-Ref and the Lane-Ref attributes and search for the element ID inside the



(a) BPMN, as from original documentation

(b) BPeX model

Fig. 1. A comparison between BPMN and BPeX hierarchical model representation.

Process element. Instead, using BPeX one has just to check whether the element node is a child of the Lane node.
The Process element is depicted with a gray background and no black boundary because it does not have a graphical representation inside a BPD. As it is possible to notice in Fig. 1(b), we also include into the model flows specifications, represented with a gray background, while in BPMN flows are disconnected from any other element. This makes them more context-free but it is not clear using XPDL where a modeler can use them or where they are defined.

Our model introduces some useful features like e.g. the hierarchical representation of BPMN elements in a diagram and the feasibility of performing queries on XML data. Our notation is fully compliant with XPDL graphical appearing notation and can describe all the data interchange between processes like BPEL does. Using BPeX it is easier to join together business processes and privacy policies, firstly because it can represent the whole set of BPMN elements. Secondly, it reproduces faithfully the diagram structure without any loss of information. Thus, it is possible to find the right position to declare a policy statement keeping privacy policies and BPs structures unaltered. This makes easier to analyze BPs (at different levels of granularity) to determine if they comply with the given privacy policy. Further information can be found on project Web page [18].

4 P3P Policy Enforcement

Currently, the main use of the P3P language is for web services providers, which can host policies in their servers leaving users to opt, using the service provided or not. Some browsers can access P3P policy document and warn users if a server policy does not accomplish the users' preferences.

BPMN (and BPeX, consequently) can easily be adopted to describe business processes whose tasks have to follow a given privacy policy. Extending the BPeX model in order to add P3P support permits users to test if a web-enabled business process is compliant with a given privacy policy. For example, a web service provider which asks user for a credit card number to perform a given task could be in contrast with the privacy policy which does not allow to ask for a personal information.

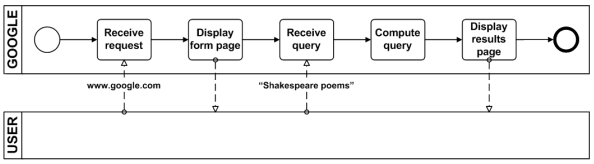


Fig. 2. BPMN representation of a user connecting to a search engine to perform a query.

In our approach, we will extend less as possible BPMN notation in order to keep the main requirements unchanged. Notice that P3P does not implement a full privacy policies tuple unlike for example the RBAC model. This is because P3P is a web-oriented standard. The main aspects we will use to extend BPMN notation are Entity, Purposes, Access, Data-group and Recipient.

For some of these elements (Entity, Data-group) we will use BPMN native attribute, extending the notation to better explain and represent the values of P3P elements. For Access element we will add a new attribute to a BPMN Process element. For the other elements we need to redefine or tune BPMN elements modifying some attributes or adding new ones. These modifications will be mapped also in BPeX to achieve a full XML linearization of BPDs with P3P statements. We have developed also some simple procedures to perform validation tests between BPeX and P3P policies trees. Some examples will be shown in the following using W3C XPath queries.

4.1 Motivating Example

The example we introduce in this section sketches the environment we are interested in. We start considering a web-oriented business process that we represent with BPMN. Then we translate the BPD into a BPeX representation. In Section 4.2 we illustrate how to integrate P3P policies and BPeX documents. Finally, we present some excerpts of BPeX code enriched with privacy policies and the algorithms to enforce the process policy.

For our running example we use a classical scenario of a user connecting to a search engine to perform a query. For the sake of clarity, we opt for Google and its privacy policies freely available on-line at Google Privacy Center[1]. Figure 2 shows the BPMN model of the Business Process we choose to investigate and Listing 1.3 (available in Appendix A) is part of its BPeX linearization. The text of the search engine privacy policy has been taken from the on-line version and the Listing 1.4 (Appendix A) is related to its P3P form.

4.2 P3P Representation Inside BPeX Code

We now summarize the formalisms we use to represent P3P clauses inside the BPeX code, investigating more in detail as possible each correspondence.

[1] http://www.google.com/privacypolicy.html

Entity. This element refers the legal entity making the representation of the privacy practices. There are only two elements in BPMN that can be used to map the Entity: the BPD and the Pool. We can not use the Process element because there may be Pools without a related Process (Black Boxes). Nonetheless, also in these cases a Pool represents a subject involved in the BP. Between BPD and Pool elements we choose to use the latter, because a BPD is a set of all the processes pertaining the BP, while a Pool represents a single actor (i.e., a single `Entity`). P3P binds some values to be present in a policy. A P3P `Entity` must hold the `orgname` attribute and one of the following categories of information: `postal`, `telephone`, `email`, `URI`. For a sake of simplicity we extend the `Name` attribute of Pools (i.e., `Pool/Name`) adding a new sub-tree starting with the `<P3PExension>` node, father of an `<Entity>` node. The Entity node imitates the P3P `Entity` nodes structure, with the same constraints. The new node `P3PExtension/Entity/orgname` substitute the old Pool Name. To add the same P3P `Entity` subtree to the `Pool/Name` attribute makes easier to compare values and enforces this policies element: there should be a direct correspondence between the two nodes structures, as depicted in Figure 4 in the Appendix A. This is a true advantage in using BPeX model respect to the original BPMN proposal, because with the latter it is not possible to make a comparison node-to-node.

Access. The P3P Access element represents the ability of the individual to view identified data and address questions or concerns to the service provider [3]. In this case, BPMN does not have an element near to the Access, but each Pool holding activities and flows has also a relationship with one Process. We add to Process element a new attribute `<P3PExtension>` having as child the `<ACCESS>` element. Possible values of `<ACCESS>` element are those provided by P3P standard.

Purposes. Every 'Common Graphical Object' (i.e., all the graphical objects that may appear in a BPD) have a `Categories` attribute, defined as follows: "The modeler MAY add one or more defined categories that can be used for purposes such as reporting and analysis". Thus, the use of this element as a container for the P3P Purposes element does not require any further adjustment unless a boolean attribute, named `IsP3PPurpose`, to better define the purposes domain. All the BPMN elements except for BPD and Process (that have not a graphical representation) have the Categories attribute, so we can define, for every element, which purpose it is designed for.

Data-group. This is the most critical issue because P3P is very rich of details about data while BPMN provides only an Artifact named `DataObject` to represent all kind of data. To describe as good as possible the information exchanged through the activities and the flows of a BP we use the Name attribute of the DataObject element to specify what kind of data an entity or an user is working on. As well as we have done to map Entity element into the `Pool/Name` attribute, we extend the `DataObject/Name` attribute with a `<P3PExtension>` node, containing the trees provided by P3P Data-Group element. In addiction, BPMN DataObjects have a `RequiredForStart` boolean attribute, that can be used to map the P3P `always`, `opt-in` and `opt-out` values for purposes and recipients.

Recipient. This element contains one or more recipients of collected data. It is the legal entity, or domain, where data may be distributed. The mapping of the `Recipient` element is a bit more complex. The best place to attach the Recipient data is a `Message Flow` (which represents the messages exchanged between different Pools – and, thus, different Entities). Unfortunately MessageFlows do not have a direct attribute where to specify the Recipient constraints. It is unnecessary to control messages exchanged inside the same Pool, between different Lanes: we suppose that an Entity can freely share data with its offices or internal employees. Again, we are not interested to investigate where data come from (typically, in BPMN diagrams, through Message Flows) – it is the sender Entity which have to adhere to its privacy policy. We need to ensure that data collected from an enterprise are the same of those declared in its privacy policy. What we can not express in BPMN is the affiliation domain of the messages targets. P3P does not need to know the target entity data, but only if the target, for example, has the same privacy policies or if it is the legal entity following the practices, and so forth. To add this kind of information, we extend the `Target` node of a `Message Flow` with an attribute `P3PRecipient` expressing the P3P values provided for the `Recipent` element.

5 The Compliance Checking Procedures

Our goal is to check whether a BPMN diagram, representing a web-enabled business process, is compliant with a P3P privacy policy. Thus, as discussed in the previous section, we have enriched the BPeX XML-based BPMN representation with some P3P-like attributes. Now, we provide checking procedures in order to verify such compliance. The tests we are interested in focus on the presence of the same attributes either in BPeX and in P3P trees. P3P notation is not used to express the values collected for each instance of the service provided. Thus, the tests will not cover the correspondence between the values which can be performed only when a process has been executed through log analysis or using a monitor.

We start assuming that each *Pool* represents an *Entity*, and thus we make the tests on *Entity* and *Access* between the *Pool* attributes and respectively `POLICY/ENTITY` and `POLICY / ACCESS` attributes. All the other tests are performed for each P3P `STATEMENT` clause, and focus on: what kind of data the process works on, how the process uses collected data, and with whom an entity shares collected data. In general it is not true that every `STATEMENT` element corresponds to one single Pool: a Pool references one Policy but it may have more than one Statement.

The diagram shown in Fig. 3 relates to our Google example: there is one Policy with altogether four Data-Ref elements, three Purposes and two different Recipients. This example shows how different Statements can act using different triples `<Data-Groups, Purposes, Recipients>`. P3P standard specifies that each Statement must hold one Data-Group node and may have more than one Purpose or Recipient expressed. In our example, the *Statement A* uses all the four `<DATA-REF>` values as Data-Group for the Purposes `<admin><develop>` sharing data with Recipient `<ours>`; the *Statement B* instead uses only two of the `<DATA-REF>` elements as Data-

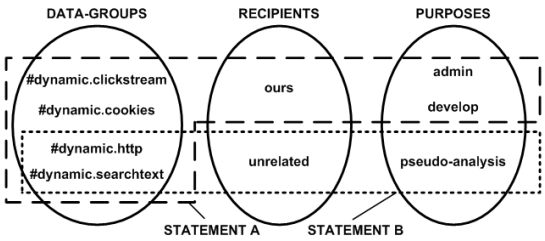


Fig. 3. Data-Groups, Purposes, Recipients relationships.

Group for the Purpose `<pseudo-analysis>` disclosing data to `<unrelated>` Recipients.

For the Statements verification we ensure firstly the correctness of the three fields separately and then the accordance between them and against the referential P3P policy.

5.1 Policies Enforcement

We introduce now a high level description of the checking procedures. For a sake of simplicity, the procedures presented here do not consider the strings manipulation needed to extract the P3P clauses from the attributes value. This aspect will be shown later on, when we will illustrate an XPath implementation example of one of these procedures. Considering that each one of the algorithms verifies a different aspect of the policy, then all of these have to be executed to enforce the privacy policy in its entirety. For a matter of space, Access, Purposes, Data-Group and Recipient listings can be found in Appendix A.

ENTITY Verification. The Listing 1.1 shows the algorithm to enforce the policy of a business process focusing on the Entity verification. This control applies on every Pool (row 1). The first condition (row 2) verifies if the `P3PExtension` node, child of `Pool/Name`, exists: if not, an error occurs (moreover, this implies that the diagram is not compliant with BPMN specifications, because the new node `Name / P3PExtension / orgname` corresponds to the original `Name` value). The core of the algorithm compares the `P3PExtension/ENTITY` subtree with the `P3P:POLICY / ENTITY` one (row 5) like in Fig. 4.

```
foreach (Pool/Name PN ∈ BPD) do {                                    1
  if (PN/P3PExtension/ENTITY == ∅)                                   2
   then ''Error''                                                    3
  elseif (PN/P3PExtension/ENTITY ≠ P3P:POLICY/ENTITY)                4
   then ''Error'';                                                   5
  else ''OK''; }                                                     6
```

Listing 1.1. The ENTITY enforcement algorithm.

ACCESS Verification. The Access verification algorithm (see Listing 1.6) is quite similar to the Entity one. It differs from the latter especially for the first condition (row 1) in which there is a check to assure that the Pool is not a Black Box: otherwise, it can not have an Access attribute because the content of the Pool is hidden and users can not access their data.

PURPOSES Verification. In this case (see Listing 1.5), firstly we consider only a subset of the all Common Graphical Objects. We argue that Swimlanes, Group and Text Annotation can not have a related Purpose. Secondly, we check if the Categories element has the required boolean attribute. Finally, we compare Categories children with all the Purpose children. Notice that between POLICY and PURPOSES at row 7 there are two slashes '`//`' to show, using the XPath syntax, that Purposes nodes are not direct Policy children but they are Policy descendants through the Statement node.

DATA-GROUP Verification. Similarly to Entity and Access verification, to enforce Data-Group elements requires to check if the P3PExtension node has been declared and successively if its values fall into the set of every Statement's Data-Group (Listing 1.7).

RECIPIENT Verification. To determine if MessageFlows are compliant with their related policies, it is necessary to control if the value of the P3PRecipient attribute is one of those declared as policy Recipient.

Listings 1.5, 1.7 and 1.8 need to be executed at Statement level, while Listings 1.1 and 1.6 at Policy level. To enforce the whole process against a privacy policies, we need to evaluate the latter once and the former for each Statement. If all test pass we can claim that the process is compliant with the privacy policy.

For a matter of space, we give now only one example of XPath translation (Listing 1.2). The BPeX code (which represents the process) and the policy code are marked with different namespaces. In the example we omit the `bpex` namespace for a reason of space. We employ to compare two node-sets the XPath 2.0 function `fn:deep-equal` which assesses whether two sequences are deep-equal to each other. To be deep-equal, they must contain items that are pairwise deep-equal; and for two items to be deep-equal, they must either be atomic values that compare equal, or nodes of the same kind, with the same name, whose children are 'deep-equal'. For a sake of simplicity this code uses the P3P 1.0 node structure `POLICIES/POLICY/ENTITY/DATA-GROUP`.

```
(//Pools/Name/P3PExtension/ENTITY)                                    1
 then fn:deep-equal(//Pools/Name/P3PExtension/ENTITY,                 2
                    p3p:POLICIES/p3p:POLICY/p3p:ENTITY)               3
\vspace{-8mm}                                                        4
```

Listing 1.2. The XPath version of the Entity algorithm.

6 Conclusions

In this paper, we proposed a new XML-based notation called BPeX allowing users to represent all BPMN elements in a hierarchical tree-based structure. We introduced an abstract representation of the BPeX notation, giving a close look to the data model representation and to the flow relationships. Then, we defined the XML-Schema and the XML linearizations of the BPeX data- and flow-model. Finally, we extended BPeX notation with the support for P3P policies. We showed the feasibility to query the BPeX representation of a BPD extended with P3P statements, in order to verify its adherence to a given P3P privacy policy specification. Please, refer to project web-site for further comparisons, case studies and related works.

References

1. WfMC: Workflow Management Coalition - Terminology & Glossary. On WfMC website (1999) WFMC-TC-1011. http://www.wfmc.org/standards/docs.htm.
2. White, S.A.: Business Process Modeling Notation - OMG Final Adopted Specification. On BPMN website (2006) http://www.bpmn.org.
3. Cranor, L., Dobbs, B., Hogben, G., Marchiori, M., Schunter, M., et al.: The Platform for Privacy Preferences 1.1 (P3P1.1) Specification (2006) http://www.w3.org/TR/P3P11/.
4. Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., et al.: Business Process Execution Language for Web Services - Version 1.1. IBM website (2003) http://www-128.ibm.com/developerworks/library/specification/ws-bpel/.
5. WfMC: Process Definition Interface – XML Process Definition Language. WfMC website (2005) WFMC-TC-1025. http://www.wfmc.org/standards/docs.htm.
6. Swenson, K.: The BPMN-XPDL-BPEL value chain. In blog "Go Flow" (2006) http://kswenson.wordpress.com/2006/05/26/bpmn-xpdl-and-bpel/.
7. Recker, J., Mendling, J.: On the Translation between BPMN and BPEL: Conceptual Mismatch between Process Modeling Languages. In: Proceedings 18th International Conference on Advanced Information Systems Engineering. Proceedings of Workshops and Doctoral Consortiums, Latour, Thibaud and Petit, Michael, Eds. (2006) 521–532
8. Ouyang, C., van der Aalst, W.M., Dumas, M., Hofstede, A.H.M.t.: From Business Process Models to Process-oriented Software Systems: The BPMN to BPEL Way. Technical Report BPM-06-27, BPM Center (2006) http://www.bpmcenter.org.
9. Ouyang, C., van der Aalst, W.M., Dumas, M., Hofstede, A.H.M.t.: Translating BPMN to BPEL. Technical Report BPM-06-02, BPM Center (2006) http://www.bpmcenter.org.
10. Mendling, J., Neumann, G., Nüttgens, M.: A Comparison of XML Interchange Formats for Business Process Modelling. In Feltz, F., Oberweis, A., Otjacques, B., eds.: EMISA. Volume 56 of LNI., GI (2004) 129–140
11. Mendling, J., de Laborda, C.P., 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.: Wissensmanagement (LNCS Volume). Volume 3782 of Lecture Notes in Computer Science., Springer (2005) 570–579
12. Mendling, J., de Laborda, C.P., Zdun, U.: Towards Semantic Integration of XML-based Business Process Models. In Althoff, K.D., Dengel, A., Bergmann, R., Nick, M., Roth-Berghofer, T., eds.: Wissensmanagement, DFKI, Kaiserslautern (2005) 513–517
13. Karjoth, G., Schunter, M.: A Privacy Policy Model for Enterprises. In: CSFW, IEEE Computer Society (2002) 271–281
14. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: An XPath-based Preference Language for P3P. In: 12th International World Wide Web Conference. (2003)
15. Bertino, E., Crampton, J., Paci, F.: Access Control and Authorization Constraints for WS-BPEL. icws 0 (2006) 275–284
16. Li, Y.H., Paik, H.Y., Benatallah, B., Benbernou, S.: Formal Consistency Verification between BPEL process and Privacy Policy. In: Privacy Security Trust 2006 (PST 2006), McGraw Hill (2006) 212–223
17. Mendling, J., Strembeck, M., Stermsek, G., Neumann, G.: An Approach to Extract RBAC Models from BPEL4WS Processes. In: WETICE, IEEE Computer Society (2004) 81–86
18. BPeX Project Site: http://bpex.sourceforge.net (2005)

A Appendix

```
<POOL ID=‘‘P001’’><NAME>Google</NAME>
 <LANE ID=‘‘L001’’><NAME>Google</NAME>
  <EVENT ID=‘‘E001’’ EventType=‘‘Start’’>
   <NONE/>
  </EVENT>
  <TASK ID=‘‘T001’’ Name=‘‘Receive request’’/>
  <TASK ID=‘‘T002’’ Name=‘‘Display form page’’/>
  <TASK ID=‘‘T003’’ Name=‘‘Receive query’’/>
  <TASK ID=‘‘T004’’ Name=‘‘Compute query’’/>
  <TASK ID=‘‘T005’’ Name=‘‘Display results page’’/>
  <EVENT ID=‘‘E002’’ EventType=‘‘End’’>
   <NONE/>
  </EVENT>
 </LANE>
 <SEQUENCEFLOW ID=‘‘SF001’’>
  <SOURCE>E001</SOURCE>
  <TARGET>T001</TARGET>
 </SEQUENCEFLOW>
 <SEQUENCEFLOW ID=‘‘SF002’’>
  <SOURCE>T001</SOURCE>
  <TARGET>T002</TARGET>
 </SEQUENCEFLOW>
 <SEQUENCEFLOW ID=‘‘SF003’’>
  <SOURCE>T002</SOURCE>
  <TARGET>T003</TARGET>
 </SEQUENCEFLOW>
 <SEQUENCEFLOW ID=‘‘SF004’’>
  <SOURCE>T003</SOURCE>
  <TARGET>T004</TARGET>
 </SEQUENCEFLOW>
 <SEQUENCEFLOW ID=‘‘SF005’’>
  <SOURCE>T004</SOURCE>
  <TARGET>T005</TARGET>
 </SEQUENCEFLOW>
 <SEQUENCEFLOW ID=‘‘SF006’’>
  <SOURCE>T005</SOURCE>
  <TARGET>E002</TARGET>
 </SEQUENCEFLOW>
</POOL>
<POOL ID=‘‘P002’’><NAME>User</NAME>
</POOL>
<MESSAGEFLOW ID=‘‘MF001’’>
 <SOURCE>P002</SOURCE>
 <TARGET>T001</TARGET>
 <MESSAGE>www.google.com</MESSAGE>
</MESSAGEFLOW>
<MESSAGEFLOW ID=‘‘MF002’’>
 <SOURCE>T002</SOURCE>
 <TARGET>P002</TARGET>
 <MESSAGE/>
</MESSAGEFLOW>
<MESSAGEFLOW ID=‘‘MF003’’>
 <SOURCE>P002</SOURCE>
 <TARGET>T003</TARGET>
 <MESSAGE>‘‘Shakespeare poems’’</MESSAGE>
</MESSAGEFLOW>
<MESSAGEFLOW ID=‘‘MF004’’>
 <SOURCE>T005</SOURCE>
 <TARGET>P002</TARGET>
 <MESSAGE/>
</MESSAGEFLOW>
</BPD>
```

Listing 1.3. The BPeX linearization of the BPMN diagram.

```
xmlns="http://www.w3.org/2002/01/P3Pv1">
 <POLICY name="Google_Example_Policy"
  discuri="http://www.google.com/privacypolicy.html"
  xml:lang="en">
  <ENTITY>
   <EXTENSION>
    <p3p11:data-group>
     <p3p11:datatype>
      <p3p11:business>
       <p3p11:orgname>Google Inc.</p3p11:orgname>
       <p3p11:contact-info>
        <p3p11:postal>
         <p3p11:street>1600 Amph.Parkway</p3p11:street>
         <p3p11:city>Mountain View</p3p11:city>
         <p3p11:state>CA</p3p11:state>
         <p3p11:postalcode>94043</p3p11:postalcode>
         <p3p11:country>USA</p3p11:country>
        </p3p11:postal>
       </p3p11:contact-info>
      </p3p11:business>
     </p3p11:datatype>
    </p3p11:data-group>
   </EXTENSION>
   <DATA-GROUP>
    <DATA ref="...">for backward compatibility</DATA>
   </DATA-GROUP>
  </ENTITY>
  <ACCESS><nonident/></ACCESS>
  <STATEMENT>
   <PURPOSE><admin/><develop/></PURPOSE>
   <RECIPIENT><ours/></RECIPIENT>
   <RETENTION><stated-purpose/></RETENTION>
   <DATA-GROUP>
    <DATA ref="#dynamic.clickstream"/>
    <DATA ref="#dynamic.http"/>
    <DATA ref="#dynamic.searchtext"/>
    <DATA ref="#dynamic.cookies"/>
   </DATA-GROUP>
  </STATEMENT>
  <STATEMENT>
   <NON-IDENTIFIABLE/>
   <PURPOSE><pseudo-analysis/></PURPOSE>
   <RECIPIENT><unrelated></RECIPIENT>
   <RETENTION><stated-purpose/></RETENTION>
   <DATA-GROUP>
    <DATA ref="#dynamic.http"/>
    <DATA ref="#dynamic.searchtext"/>
   </DATA-GROUP>
  </STATEMENT>
 </POLICY>
</POLICIES>
```

Listing 1.4. The P3P form of the Google Privacy Policy.

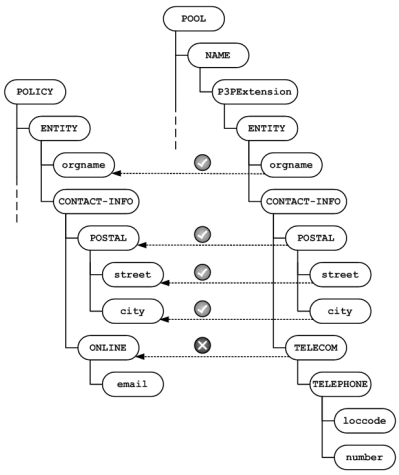


Fig. 4. A comparison between P3P and BPeX Entity elements.

```
CommonGraphicalObjects; CGO* := CGO \ (Swimlanes,                          1
Group, TextAnnotation); foreach (Pool P ∈ BPD) do {                        2
 foreach (CGOElement ∈ CGO*) do {                                          3
  if (CGOElement/Categories@IsP3PPurpose == ∅)                             4
   then ‘‘Error’’                                                          5
  elseif (CGOElement/Categories ⊈ P3P:POLICY//PURPOSES)                    6
   then ‘‘Error’’                                                          7
  else ‘‘OK’’; } }                                                         8
```

Listing 1.5. The PURPOSES enforcement algorithm.

```
foreach (Pool/Process PP ∈ BPD | PP ≠ ∅) do {                              1
  if (PP/P3PExtension/ACCESS == ∅) then ‘‘Error’’;                         2
  elseif (PP/P3PExtension/ACCESS ≠ P3P:POLICY/ACCESS)                      3
   then ‘‘Error’’                                                          4
  else ‘‘OK’’; }                                                           5
```

Listing 1.6. The ACCESS algorithm.

```
foreach (DATAOBJECT DO ∈ BPD) do {                                         1
 if (DO/NAME/P3PExtension == ∅) then ‘‘Error’’                             2
  elseif (DO/NAME/P3PExtension ⊈                                           3
                    P3P:POLICY/STATEMENT/DATA-GROUP)                       4
   then ‘‘Error’’                                                          5
  else ‘‘OK’’; }                                                           6
```

Listing 1.7. The DATA-GROUP enforcement algorithm.

```
(MESSAGEFLOW MF ∈ BPD) do {                                                1
 if (MF/Target@P3PRecipient == ∅) then ‘‘Error’’                           2
 elseif (MFM/Target@P3PRecipient ⊈                                         3
      P3P:POLICY/STATEMENT/RECIPIENT) then ‘‘Error’’                       4
 else ‘‘OK’’; }                                                            5
```

Listing 1.8. The RECIPIENT enforcement algorithm.