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
{michele.chinosi,alberto.trombetta}@uninsubria.it

The increased interest around business processes management and modeling techniques has brought many organizations to make significant investments in business process modelling projects. One of the most recent proposal for a new business process modelling technique is the Business Process Modelling 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 investigating. 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.*

Keywords: Business Processes, Privacy Policies, Business Process Modelling Notation, Platform for Privacy Preferences, Compliance Check.

Classification: H.1 Models and Principles; I.6 Simulation and Modelling; K.4.1 Public Policy Issues.

1. INTRODUCTION

The ever-increasing interest around business processes management and modelling techniques has brought many organizations to make significant investments in business process modelling 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* (WfMC, 1999). As such, it introduces methods, tools and techniques to support the development and the analysis of operational business processes. Inside this context, business process modelling techniques and languages are of absolute relevance. Again, WfMC defines business process modelling as *the time period when manual and/or automated (workflow) descriptions of a process are defined and/or modified electronically* (WfMC, 1999). One of the most recent proposal for a business process modelling technique is Business Process Modelling Notation (BPMN), adopted as standard by OMG (White, 2006).

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 recent years, several efforts have been

Manuscript received: 23 April 2008
Communicating Editor: Eduardo Fernandez-Medina Paton

made in order to provide mechanisms for expressing (and in some cases, enforcing) privacy policies protecting such data. One of the most well-known efforts in such direction is the P3P privacy policy description language (Cranor, Dobbs, Hogben, Marchiori, Schunter *et al*, 2006). P3P permits representation of privacy policies in an XML tree, based 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 (Andrews, Curbera, Dholakia, Goland, Klein, Leymann, Liu, Roller *et al*, 2003) or XPDL (WfMC, 2005; Swenson, 2006). 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 (Recker and Mendling, 2006; Ouyang, van der Aalst, Dumas and ter Hofstede, 2006a; Ouyang, van der Aalst, Dumas and ter Hofstede, 2006b). 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 (WfMC, 2005; Swenson, 2006). 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.

This work is organized as follow: after introducing some related efforts, in Section 3 a brief summary of our proposal BPeX is presented. In Section 4 a motivating example is described firstly using BPMN graphical notation and secondly using the BPeX metamodel and its XML-Schema serialization. In Section 5 some compliance checking procedures are provided in order to evaluate the example policy. Right before introducing some conclusions and further works, in Section 6 a graphical aiding support for privacy policies is suggested.

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 (Mendling, Neumann and Nüttgens, 2004a; Mendling, de Laborda and Zdun, 2005a; Mendling, de Laborda and Zdun, 2005b),

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 (Karjoth and Schunter, 2002; Agrawal, Kiernan, Srikant and Xu, 2003; Bertino, Crampton and Paci, 2006) even though the first work proposing an algorithm to verify P3P policies on BPEL4WS tree was published in 2006 (Li, Paik, Benatallah and Benbernou, 2006). Moreover, in Mendling, Strembeck, Stermsek and Neumann (2004b) an approach to extract RBAC models from BPEL4WS processes for the role in the engineering process is presented.

Due to space limitations, you can find more detailed information about P3P and BPMN on respective standard web-pages, (Cranor et al, 2006; White, 2006). 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` (BPeX Project Site, 2005). On the same web-site you can also find an extended version of this paper.

3. BPEX: A BPMN XML LINEARIZATION

BPMN defines a graphical notation without an explicit definition of the underlying 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, the behaviour) 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 (Mendling *et al*, 2004a; Mendling *et al*, 2005a; Mendling *et al*, 2005b). There are some research efforts aimed at the definition of a comprehensive model representing *all* the main features of BPMN, but all such efforts were built upon existing incomplete and/or inadequate formats. As we already mentioned, either WS-BPEL and XPDL have to consider their past structures and goals. We have chosen a clean start by looking closely into BPMN and building our model from scratch, thus obtaining a clear conceptual model (or simply model, thereon) not based on previous proposals (WS-BPEL and XPDL) and intrinsically supporting all the relevant features of BPMN. We aim to propose a complete, compact and easy-to-navigate model for BPMN which could be adopted as the nearest serialization format to store, exchange and execute BPDs, providing also the needed mechanisms to export the model towards XPDL, WS-BPEL and other formats.

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, then we proceed refining them through the definition of more precise symbol families, connected in a suitably defined hierarchy. Afterward, 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 terminology we call BPeX both the model and its XML-based version.

3.1 The BPeX Conceptual Model

To develop the new model we follow a top-down fashion methodology as the one provided by BPMN specifications, which starts from the root of a diagram definition (the *BPD* element) going through the definition of the elements and their attributes. BPMN 1.1 specifications provide some UML diagrams to better specify the relationships between elements and types. There are two main aspects to notice. First, there is no difference between elements and types: they are all defined as

elements (and represented as classes in UML diagrams). Second, all the BPMN elements (except for *BPD* which has no connections at all) are connected through UML generalizations, and this means that all the elements except for the root are extensions of other elements. But there are some orthogonal implicit connections which can be defined as *Connections-by-Types*. Some elements attributes are defined having as type another element (e.g., all the *Properties* attributes are of type *Property*, which is an element child of the *Supporting Element* element). These connections could be intended as foreign keys relationships.

We aim to minimize these connections and avoid all the missing connections (as in case of *BPD*). We analyze every element dependencies from a hierarchical point of view as well as its graphical placements constraints. The model we define should be compliant to BPMN specifications. The root of the model is the *BPD* element. Every *BPD* must have at least one *Pool* which must contain one or more *Lanes*. *Activities*, *Events* and *Gateways* must be placed inside a Lane boundaries. The *Artifacts* can be positioned everywhere inside a BPD since they can be connected to every other object. Thus they can be viewed as BPD children. All the elements defined by BPMN specifications as *Supporting Elements* in BPeX are treated as types rather than elements, simplifying the model structure. The connections between elements can be defined as *Connections-by-Semantics*, because we build a BPeX model considering the behaviour and the possible placements of the elements inside a diagram. This structure can be easily represented with a diagram as the one depicted in Figure 1. We use UML to better show differences between the BPMN original model and our proposal but for comparison purposes only. BPeX is independent from any model definition format. We use UML compositions to represent the *Connections-by-Semantics* relationships, because they adequately reflect the hierarchical positioning model of a BPD. We maintain instead the use of UML generalizations only in cases of elements extensions (like, e.g., *Events* or *Artifacts*).

Looking at the *Connecting Flow* objects, we can find them an appropriate placement inside the model. A *Sequence Flow* can connect objects inside the same *Pool*, crossing the *Lanes* boundaries but it must not cross the *Pool* boundaries. So it is reasonable to put SequenceFlows as children of *Pool* element. Thus, it becomes impossible to define a *Sequence Flow* outside a *Pool* object. *MessageFlows* must connect objects belonging to different *Pools*. We place them as children of *BPD* element. Finally, *Associations* link *Artifacts* to every object in the BPD and, thus, we put them at the same level of *Artifacts*, as children of *BPD*.

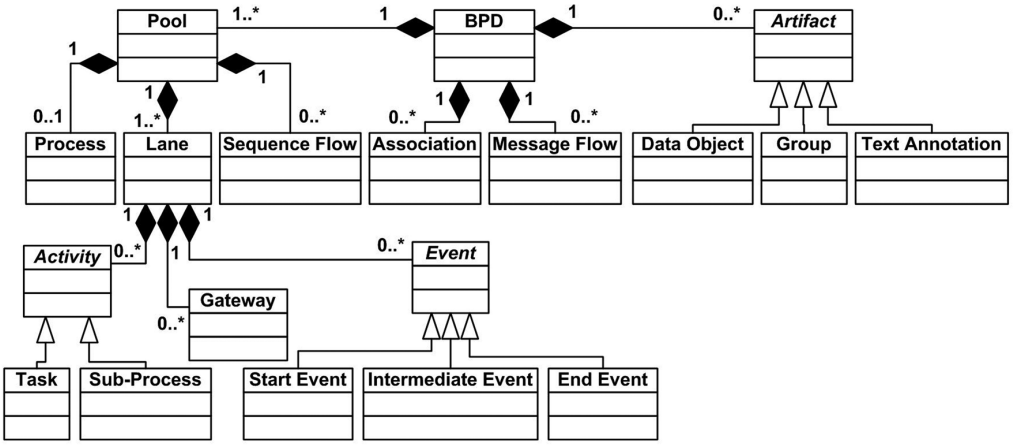

Figure 1: A high-level view of the BPeX conceptual model layout

```
//Lane[//Task/@Id = 10]/@Name                                    1
                                                                 2
Result: /BPD[1]/Pool[2]/Lane[1]/@Name - Lane-0                   3
```

Listing 1: The XPath query needed to find a Lane starting from one of its contained elements using BPeX

One of the most immediate advantages of using a full hierarchical model is, e.g., the complexity of checking what *Lane* an element (an *Event*, an *Activity* or a *Gateway*) belongs to. Using the BPMN model, one has to control the values of the `Pool` and the `Lanes` attributes and search for the *ID* of the element inside the `GraphicalElements` attribute of the *Process* element. Using XPDL the query is too complex. Instead, as it is possible to see in List. 1, using BPeX one has just to check whether the element node is a child of the *Lane* node. This new model enhances the constraints between the objects and simplifies the main steps for a process analysis. Moreover, we clean the model from weak and confusing connections.

3.2 XML Linearization

Once we defined the BPeX data model and the relationships between the objects, we start serializing it using the XML-Schema language, as for WS-BPEL and XPDL. The XML-Schema language applies in a very natural way to serialize strict hierarchical data models, providing also a good types support to define attributes as close as possible to their original BPMN definitions. Furthermore, the use of XML-Schema eases the integration between BPeX and the other XML-based standards and the extension with new features.

To enforce the referential integrity of our model we enrich the XML serialization with couples of `xsd:key` and `xsd:keyref`. As a matter of fact, `xsd:ID` and `xsd:IDRef` are not enough to guarantee that all the references are correct. There is not an easy way to differentiate *ID* values to ensure that every element type has its own pattern-based *ID* family (i.e., the *ID* family of *Lanes* is different from other *ID* families). This implies that, e.g., the `Lane` attribute of a *Task* – which needs to have the *ID* of a *Lane* as value – could have as value the *ID* of one *StartEvent*. Syntactically the document is valid. However, we need a more strict control on the values a reference should have. An `xsd:key` can assign a unique name (the key) to every element or path inside an XML-tree. Similarly an `xsd:keyref` connects an `xsd:key` value to one specific element or path. Thus, we can assure that the `Lane` attribute of a *Task* object will have as value one of the right *Lane ID* values. We use `xsd:key/xsd:keyref` also to declare which attributes an object must have according to the value another attribute gets when it is instantiated. For instance, the *StartEvent* object changes the set of its attributes according to the value of its `Trigger` attribute. If the `Trigger` attribute value is *Message*, then the *StartEvent* should have `Message` and `Implementation` as attributes and no other attributes like `TimeDate` or `ProcessRef`. To our best knowledge, the most suitable way the other serialization formats implement this characteristic is to define an element for every set of allowed attributes, introducing more complexity, redundancy and going far from the original BPMN definition.

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 a 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.

In our approach, we will extend the least 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 (*http://www.google.com/ privacypolicy.html*). Figure 2 shows the BPMN model of the Business Process we chose to investigate and Listing 12 (available in the Appendix) is part of its BPeX linearization. The text of the search engine privacy policy has been taken from the online version and the Listing 11 (Appendix) is related to its P3P form. This example takes care of most of the Google Privacy Policy conditions. For example this process shows the cookies exchange between Google and the user's browser or the data that are stored in a log file. The process models the alternatives that can be followed depending on the registration status of the user to the Google services. In some cases in this paper we will show some excerpts taken from this process. A bigger version of Figure 2 can be found in the Appendix.

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 is possible for each correspondence.

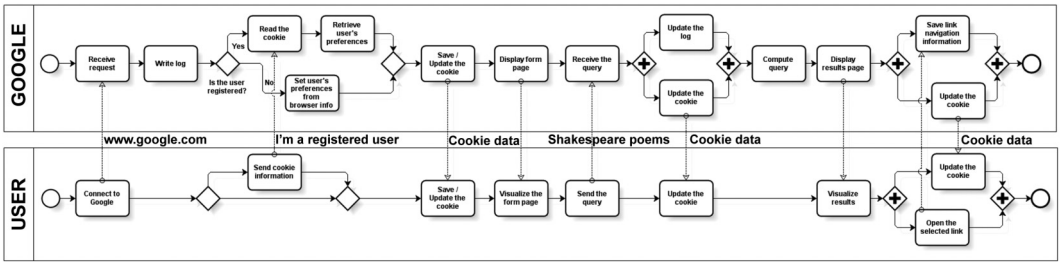

Figure 2: BPMN representation of a user connecting to a search engine to perform a query

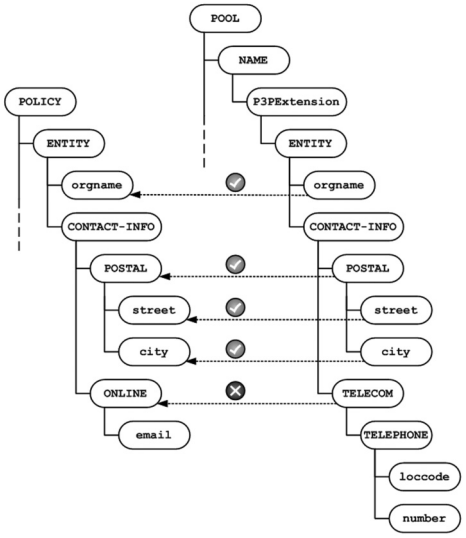

Figure 3: A comparison between P3P and BPeX Entity elements

Entity. This element refers to 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 cannot 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 chose 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 the 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 it easier to compare values and enforces this policies element: there should be a direct correspondence between the two nodes structures, as depicted in Figure 3. This is a true advantage in using BPeX model in 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 (Cranor *et al*, 2006). In this case, BPMN does not have an element near to the Access, but each Pool holding activities and flows also has a relationship with one Process. We add to the 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 in 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 a user is working on. 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 in investigating where data comes 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 as those declared in its privacy policy. What we cannot 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 `Recipient` 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 Figure 4 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

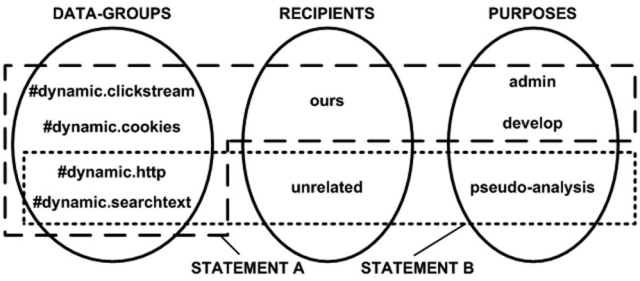

Figure 4: Data-Groups, Purposes, Recipients relationships

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-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 the 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.

ENTITY verification. The List. 2 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 4) like in Figure 3.

ACCESS verification. The Access verification algorithm (see List. 3) 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

```
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 2: The ENTITY 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 3: The ACCESS algorithm

assure that the Pool is not a Black Box: otherwise, it cannot have an Access attribute because the content of the Pool is hidden and users cannot access their data.

PURPOSES verification. In this case (see List. 4), firstly we consider only a subset of all the Common Graphical Objects. We argue that Swimlanes, Group and Text Annotation cannot 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 (List. 5).

RE*CIPIENT 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 (List. 6).

Listings 4, 5 and 6 need to be executed at Statement level, while List. 2 and 3 at Policy level. To enforce the whole process against privacy policies, we need to evaluate the latter once and the former for each Statement. If all tests pass we can claim that the process is compliant with the privacy policy.

For a matter of space, we now give only a few examples of XPath translations (Listings 7, 8, 9). 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

```
CGO := CommonGraphicalObjects;                                                    1
CGO* := CGO \ (Swimlanes, Group, TextAnnotation);                                 2
foreach (Pool P ∈ BPD) do {                                                       3
 foreach (CGOElement ∈ CGO*) do {                                                 4
  if (CGOElement/Categories@IsP3PPurpose == ∅)                                    5
   then ''Error''                                                                 6
  elseif (CGOElement/Categories ⊈ P3P:POLICY//PURPOSES)                           7
   then ''Error''                                                                 8
  else ''OK''; } }                                                                9
```

Listing 4: The PURPOSES enforcement 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 5: The DATA-GROUP enforcement algorithm

```
foreach (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 6: The RECIPIENT enforcement algorithm

```
if (//Pool/Name/P3PExtension/ENTITY)                                    1
 then fn:deep-equal(//Pool/Name/P3PExtension/ENTITY,                     2
                p3p:POLICIES/p3p:POLICY/p3p:ENTITY)                      3
```

Listing 7: The XPath version of the Entity algorithm

```
let $PP := //Process[@ID = //Pool@ProcessRef]                           1
if ($PP/P3PExtension/ACCESS)                                            2
 then fn:deep-equal($PP/P3PExtension/ACCESS,                            3
                p3p:POLICIES/p3p:POLICY/p3p:ACCESS)                      4
```

Listing 8: The XPath version of the Access algorithm

```
if (//DataObject/Name/P3PExtension)                                     1
 then some $dg in p3p:POLICIES/p3p:POLICY/p3p:STATEMENT/p3p:DATA-GROUP   2
  satisfies $dg = //DataObject/Name/P3PExtension                        3
```

Listing 9: The XPath version of the Data Group algorithm

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'. We also use the `some...satisfies` construct to better express the subset relationship, as in List. 9. For the sake of simplicity this code uses the P3P 1.0 node structure `POLICIES/POLICY/ENTITY/DATA-GROUP`.

6. GRAPHICAL REPRESENTATION INSIDE THE MODEL

We provide an easy way to aid users to identify privacy policies adherence. We extend the graphical layout of BPMN elements adding three simple markers over every element involved in a privacy policy. The three markers are shown in Figure 5, a simplified version of the running example, and represent the complete adherence of an element to its relative privacy policy constraints, a warning message in case of a partial constraints compliance and an alert message if the element does not meet the privacy policy requirements.

The use of these signs is coded inside the XML linearization using a simple string attribute for every element affected from the P3P code. In List. 10 it is possible to see an example for Tasks.

Using the motivating example, assume that the search engine changes its privacy policy but not the process. The new provided privacy policy denies the search engine to collect any navigation information for further analysis. As it is possible to see in Figure 6, the tasks Receive Request and Receive Query are marked with a warning sign, because they could collect some not allowed data during the navigation and querying activities.

Figure 5: From the left: complete, partial or missing privacy policies compliance

```
<TASK ID=''T001'' Name=''Receive request'' IsP3PCompliant=''Warning''/>          1
<TASK ID=''T002'' Name=''Display form page'' IsP3PCompliant=''Yes''/>            2
<TASK ID=''T003'' Name=''Receive query'' IsP3PCompliant=''Warning''/>            3
<TASK ID=''T004'' Name=''Compute query'' IsP3PCompliant=''Yes''/>                4
<TASK ID=''T005'' Name=''Display results page'' IsP3PCompliant=''Yes''/>         5
```

Listing 10: An example of signs coded inside the BPeX XML linearization

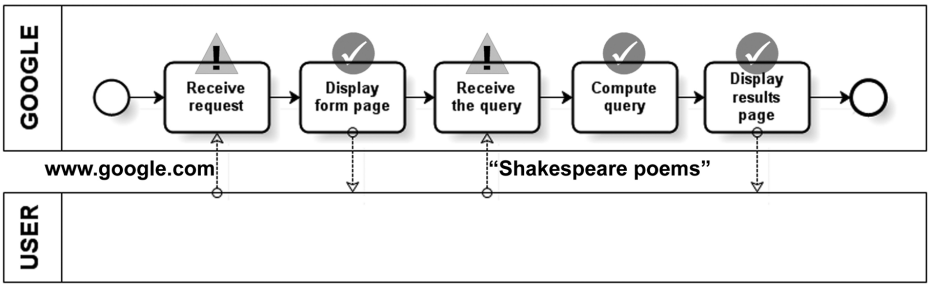

Figure 6: From the left: complete, partial or missing privacy policies compliance

7. CONCLUSIONS AND FURTHER WORKS

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 at 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 and a graphical way to help users find errors or partial adherence of a process with respect to its privacy policy. Please, refer to project web-site for further comparisons, case studies and related works.

We aim to extend the graphical notation we presented in Section 6 in order to label any relevant element of a Process. The P3P developing process stopped in 2006, and so we argue to apply these results using other technologies still supported or under development. We have shown a core integration between a BPMN serialization and some of the P3P XML-Schema elements. This integration could be extended to comprehend all the P3P elements. Also the BPeX linearization could be changed once a standard BPMN linearization is published.

REFERENCES

AGRAWAL, R., KIERNAN, J., SRIKANT, R. and XU, Y. (2003): An XPath-based preference language for P3P. In *12th International World Wide Web Conference*.

ANDREWS, T., CURBERA, F., DHOLAKIA, H., GOLAND, Y., KLEIN, J., LEYMANN, F., LIU, K., ROLLER, D. *et al* (2003): Business process execution language for web services – Version 1.1. IBM website. http://www128.ibm.com/developerworks/library/specification/ws-bpel/. Accessed April 2008.

BERTINO, E., CRAMPTON, J. and PACI, F. (2006): Access control and authorization constraints for WS-BPEL. *Proc. The IEEE International Conference on Web Services (ICWS)*, Chicago, USA, 275–284.

BPEX (2005). The BPeX Project Site. http://bpex.sourceforge.net. Accessed March 2008.

CRANOR, L., DOBBS, B., HOGBEN, G., MARCHIORI, M., SCHUNTER, M. *et al* (2006): The platform for privacy preferences 1.1 (P3P1.1) Specification. http://www.w3.org/TR/P3P11/. Accessed April 2008.

KARJOTH, G. and SCHUNTER, M. (2002): A privacy policy model for eenterprises. *15th IEEE Computer Security Foundations Workshop*, Cape Breton, Canada, 271–281. IEEE Computer Society.

LI, Y.H., PAIK, H.Y., BENATALLAH, B. and BENBERNOU, S. (2006): Formal consistency verification between BPEL process and privacy policy. *Privacy Security Trust 2006 (PST 2006)*, Toronto, Canada, 212–223. McGraw Hill.

MENDLING, J., DE LABORDA, C.P. and ZDUN, U. (2005a): Towards an integrated BPM schema: Control flow heterogeneity of PNML and BPEL4WS. In ALTHOFF, K.D., DENGEL, A., BERGMANN, R., NICK, M. and ROTH-BERGHOFER, T. (eds.) *Wissensmanagement (LNCS Volume)*, vol. 3782 of *Lecture Notes in Computer Science*, 570–579. Springer.

MENDLING, J., DE LABORDA, C.P. and ZDUN, U. (2005b): Towards semantic integration of XML-based business process models. In ALTHOFF, K.D., DENGEL, A., BERGMANN, R., NICK, M. and ROTH-BERGHOFER, T. (eds.) *Wissensmanagement*, 513–517. DFKI, Kaiserslautern.

MENDLING, J., NEUMANN, G. and NÜTTGENS, M. (2004a): A comparison of XML interchange formats for business process modelling. In FELTZ, F., OBERWEIS, A. and OTJACQUES, B. (eds.) E*MISA*, vol. 56 of *LNI*, 129–140. GI.

MENDLING, J., STREMBECK, M., STERMSEK, G. and NEUMANN, G. (2004b): An approach to extract RBAC models from BPEL4WS processes. Proc. *13th IEEE International Workshops on Enabling Technologies (WETICE 2004)*, *Infrastructure for Collaborative Enterprises*, Modena, Italy, 81–86. IEEE Computer Society.

OUYANG, C., VAN DER AALST, W.M., DUMAS, M. and TER HOFSTEDE, A.H.M. (2006a): From business process models to process-oriented software systems: The BPMN to BPEL way. Technical Report BPM-06-27. BPM Center. http://www.bpmcenter.org. Accessed April 2008.

OUYANG, C., VAN DER AALST, W.M., DUMAS, M. and TER HOFSTEDE, A.H.M. (2006b): Translating BPMN to BPEL. Technical Report BPM-06-02, BPM Center. http://www.bpmcenter.org. Accessed April 2008.

RECKER, J. and MENDLING, J. (2006): On the translation between BPMN and BPEL: Conceptual mismatch between process modelling languages. *Proc. 18th International Conference on Advanced Information Systems Engineering. Proceedings of Workshops and Doctoral Consortiums*, Luxembourg, Grand-Duchy of Luxembourg, 521–532. LATOUR, T. and PETIT, M., Eds.

SWENSON, K. (2006): The BPMN-XPDL-BPEL value chain. In blog "Go Flow". http://kswenson.wordpress.com/2006/05/26/bpmn-xpdl-and-bpel/. Accessed April 2008.

WFMC (1999): Workflow management coalition – Terminology & Glossary. WFMC-TC-1011. http://www.wfmc.org/standards/docs.htm. Accessed March 2008.

WFMC (2005): Process definition interface – XML process definition language.

WFMC-TC-1025 (2008): http://www.wfmc.org/standards/docs.htm. Accessed March 2008.

WHITE, S.A. (2006): Business process modelling notation – OMG final adopted specification. On BPMN website. http://www.bpmn.org. Accessed April 2008.

BIOGRAPHICAL NOTES

Michele Chinosi received his MS in computer science from the University of Milano in 2004 and his PhD in computer science from the University of Insubria in 2009. He is contract professor at the Computer Science and Communication Department of Insubria University, Varese, Italy. His main research topics are modelling and design methodologies for Business Processes, XMl and XML-Schema representation, privacy and security issues in business processes.

Michele Chinosi

Alberto Trombetta received his MS in computer science from the University of Milano in 1994, and his PhD in computer science from the University of Torino in 2000. He is assistant professor at the Computer Science and Communication Department of Insubria University, Varese, Italy (from 2002). His main research topics are security and privacy issues in distributed, data-intensive applications and design methodologies for business processes.

Alberto Trombetta

APPENDIX

```
<POLICIES 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 11: The P3P form of the Google Privacy Policy

```
<BPD Name=''BPeX Example''>
 <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 12: The BPeX linearization of the BPMN diagram

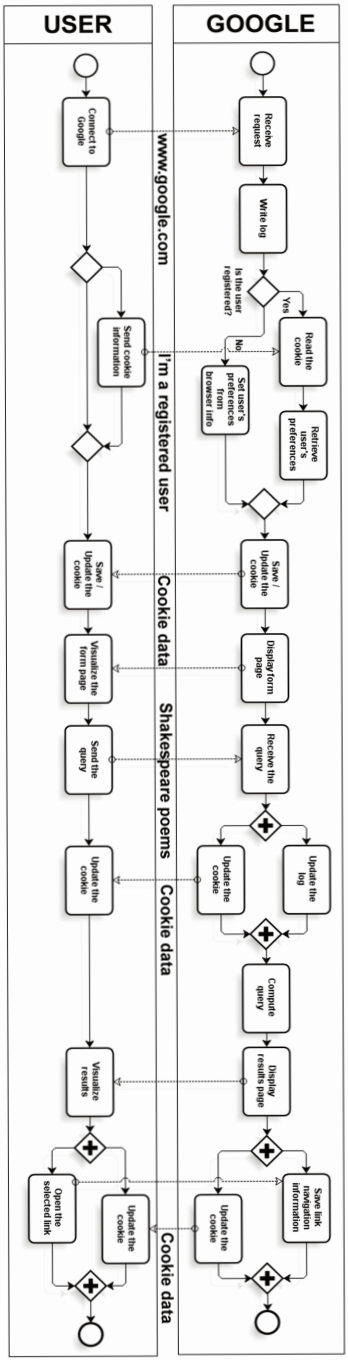

Figure 7: BPMN representation of a user connecting to a search engine to perform a query