Privacy Inspection and Monitoring Framework for Automated Business Processes

Yin Hua Li, Hye-Young Paik and Jun Chen

School of Computer Science and Engineering
University of New South Wales
Sydney, Australia
{yinhual, hpaik, jche340}@cse.unsw.edu.au

Abstract. More and more personal data is exposed to automatic and programmatic access, making it more difficult to safeguard the personal information from unauthorised access at every step. We introduce a privacy inspection and monitoring framework provides two functions: (i) the process designers can look over the privacy aspects of the automated business processes and resolve potential problems before deploying, and (ii) the obligation management technique ensures when certain actions performed on personal data, appropriate obligations are carried out.

1 Introduction

The recent emergence of Web services means that more and more business processes and their associated data are exposed to programmatic access, making it more difficult for human being to act as checkpoints that safeguard the personal information from unauthorised access at every step. This calls for a technical solution that will aid privacy officers by automatically (or semi-automatically) monitoring and detecting possible violation of privacy policies in the automated business processes.

One of the widely accepted standards in business process automation is Business Process Execution Language for Web Services (BPEL4WS, or BPEL for short) [1]. BPEL represents a programming language and its execution environment for creating composite Web services, in which interactions between autonomous entities (Web services) are defined and orchestrated. Our work focuses on the business processes written in BPEL. We summarise our contributions as follows:

- *Privacy inspection tool*: We have built a privacy inspection tool that allows the process designers to examine the privacy aspects of a BPEL process with respect to the organisational policies and resolve potential problems before deployment. This will remove the necessity for checking every activity for privacy conformance at runtime;
- *Privacy monitoring framework:* We have built an obligation management framework for a BPEL process which ensures that when certain action is performed on personal data, appropriate obligations are automatically carried out. We use Aspect-Oriented programming technique which is applied to an open source BPEL engine.

2 Privacy Model in BpelPrivacyMonitor

2.1 Privacy Data Model

Privacy Dictionaries. Different business processes collect different types of personal data for varying purposes and they involve different types of individuals or group who access the data (e.g., banking process vs. medical examination process). For this reason, we assume every organisation builds *privacy dictionaries*, in which the following information is defined in their own terms (e.g., using a language such as EPAL[1]):

- *Personal Data* (or data, for short) refers to information associated with an individual that can potentially identify the individual.
- *Data User* (or user, for short) represents a category of legal entities who can access personal data.
- *Data Usage Purpose* (or purpose, for short) describes the reason for which the data is collected and used.

Privacy Actions and Obligations. We define generic *Privacy Actions* (or actions for short) that are performed on data, namely: (i) collect – collecting the data for the first time, (ii) read – accessing the data as read-only, (iii) update – accessing the data as read-write, (iv) distribute – accessing the data to forward to partners and (v) delete – removing the data from the storage.

We also use the concept *Obligations* to dictate extra measures that must be taken when certain privacy actions occur. For example, when certain data is collected for the first time, the organisation may be obligated to log the details.

The term obligations are used in different context in security & privacy [2]. In BpelPrivacyMonitor, we observe the following measures as obligations: (i) sending a notification to the data subject (i.e., customer) about the usage of his/her data, (ii) logging the details of the usage of data, (iii) setting an alarm to automatically notify a possible over retention of data and (iv) regular reporting (e.g., monthly or yearly) to the data subject of the usage of her data. These are currently implemented in BpelPrivacyMonitor.

2.2 Privacy Policies

Based on the above abstraction, we model privacy policies as the organisation's *rights* and *obligations* on personal data. To be more precise, we define the following concepts[2]:

Definition 1 (Rights). *Let $\mathcal{U}, \mathcal{D}, \mathcal{P}$ be the privacy dictionary of users, data and purposes, respectively; and let $\mathcal{A} = \{collect, read, update, distribute, delete\}$. A right over $d (\in \mathcal{D})$, is a tuple (u, a, p), where $u \in \mathcal{U}, a \in \mathcal{A}$ and $p \in \mathcal{P}$. It is read that u can perform a on d for the purpose of p. We note it as $R_d : (u, a, p)$.*

[1] EPAL, http://www.w3.org/Submission/EPAL
[2] This model is based on our previous work [7, 5]

For example, $\mathtt{R}_{(\mathtt{email})}$: (`customer manager`, `read`, `validate transactions`) can be read as "an entity who plays a customer manager role can read the data email for the purpose validate transactions".

Definition 2 (Obligations). *Let $\mathcal{A}_{ob}$ be obligation actions, which consists of {send notification, write log, alert retention, regular report}. An obligation is a tuple (a_{ob}, μ_{ob}), where $a_{ob} \in \mathcal{A}_{ob}$ and μ_{ob} is the time property associated with a_{ob}.*

Each obligation is associated with a set of properties. One of which is time[3]. For example, write log is associated with a time property whose default value is 'immediately'. regular report also has a time property that determine how often the report should be sent to the customer (e.g., last day of each month). alert retention uses its time property to send the reminder of possible over retention of personal data (e.g., 90 days).

Definition 3 (Privacy Policy). *A privacy policy consists of a set of rights and obligation pairs. We note it as $R_d : (u, a, p) \rightarrow \{o_1, ..., o_n\}$, where R_d is the right over data d, $o_i (1 \leq i \leq n)$ is an obligation. The reading of each policy is that when a right is obtained, a set of obligations must follow.*

The example of privacy policies is given in the next section.

2.3 A running example in BPEL

Let us assume a company called BestMortgage who offers finance products such as personal, car or home loans. The company automated their home loan application process via a BPEL process homeLoanService. The process consists of a number of partners, each carrying a portion of the complete process.

The building blocks of BPEL are activities. In particular, receive, pick, invoke and reply activities provide the means of communication among the participants. Figure 1 shows the coordination of the communication.

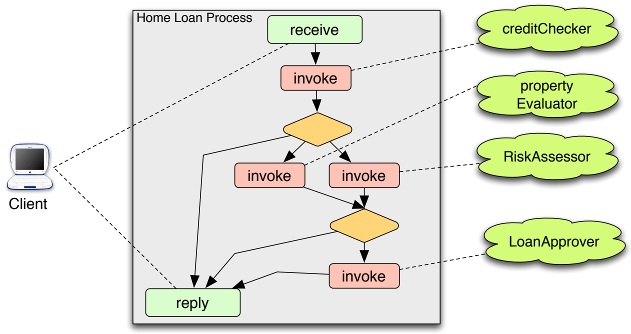


Fig. 1. homeLoanService and its partners

[3] To be generic, we associate a set of properties to an obligation. Currently, only a time property is implemented.

The process receives a home loan application from an applicant, then it invokes *CreditChecker* partner who will assess applicant's credit score. If the score is low, the application will be rejected. Otherwise, the process invokes *PropertyEvaluator* and *RiskAssessor* to obtain a property evaluation and risk evaluation, respectively. If the risk level is low, the application will be approved. Otherwise, the process invokes *LoanApprover* who will go through thorough assessment and decide whether to approve the application or not. Finally, the process replies the result to the applicant.

One thing to note is that during the communication, as shown in Fig. 2, the BPEL process sends/receives messages (referred to as variables). Some of the messages inherently contain personal data (such as income, home address, phone, etc.). The crux of the privacy issue in BestMortgage is to make sure that the personal data in the messages are exchanged in accordance with the privacy policies of the organisation. The company's privacy dictionaries and policies are as follows:

Privacy Dictionaries. The data dictionary contains {home postal, home phone, mobile phone, income}; the user dictionary includes {ours, credit checker, property evaluator, risk assessor, loan approver, public}; The purpose dictionary has {homeloan processing, risk assessment, property evaluation, send email, call customer}. The term ours refers to the organisation itself.

Privacy Policies. Here is a fragment of the privacy policies of the company which shows the company's rights on data and their associated obligations.

$\mathtt{R}_{(\mathtt{home\ postal})}$:(ours, collect, homeloan processing) $\rightarrow$ {(alert retention, 180 days)}, $\mathtt{R}_{(\mathtt{home\ postal})}$:(ours, distribute, risk assessment) $\rightarrow$ {(write log, immediately)}, $\mathtt{R}_{(\mathtt{income})}$:(risk assessor, read, risk assessment) $\rightarrow$ {(write log, immediately), (regular report, last Friday of each month)}, *... more policies ...*

3 Privacy Inspection Tool

In BpelPrivacyMonitor, before a BPEL process is deployed, it goes through what we call "privacy inspection procedure". The procedure serves two purposes: first, it enables the process designer to capture and analyse what types of personal data, user and purposes are involved in the BPEL process. The BPEL code is analysed with respect to the terms defined in the privacy dictionaries. Second, it produces a set of metadata that will be used as the basis for the monitoring procedure in the execution phase.

BpelPrivacyMonitor provides a tool Privacy Inspector, which is designed to semi-automate the inspection and metadata generation process. The tool has

```
<bpel:invoke  name="InvokeCreditScoreService" partnerLink="creditScoreLink"
    portType="ns1:creditScorePT" operation="checkCreditScore"
    inputVariable="CreditScoreRequest" outputVariable="CreditScoreReturn" />
```

Fig. 2. A BPEL activity with input/output messages (variables)

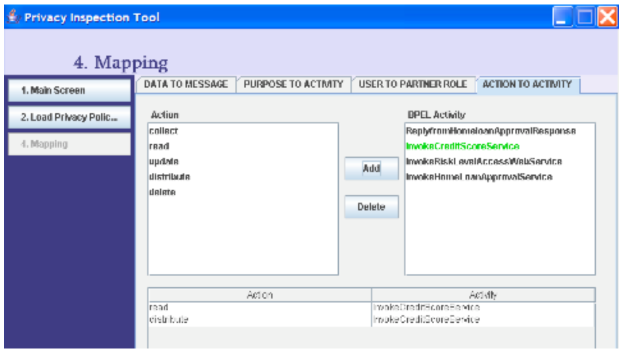
a “wizard-type” user interface, providing a step-by-step guide for the process designer.

3.1 Privacy Inspection Steps

Step 1: Associating Variables with Personal Data. The first step of the inspection involves examining the BPEL activities and their variables and marked the ones that carry personal data. At the end of this step, the identification of the activities for further inspection is done.

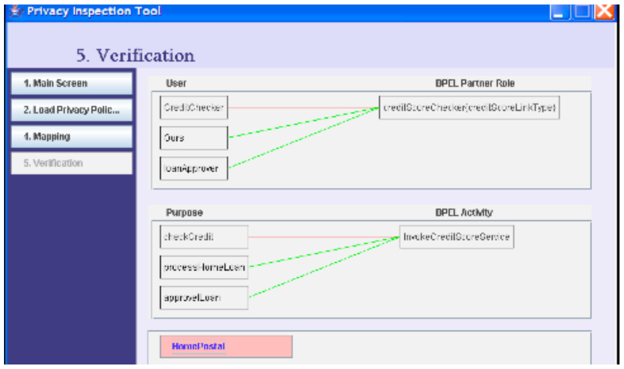
For instance, the invoke activity in Figure 2 takes creditScoreRequestMessage as input. For space reasons, we cannot present all message definitions in the BPEL code here. In our current implementation of the running example scenario, the message contains personal income and home address. Therefore, the process designer will mark the activity as being relevant to the inspection process and map its message part names to the data dictionary terms.

Step 2: Associating Activities with Actions. In the next step, the inspection tool presents, on the right hand side, the activities carried over from Step 1. The list of actions from the privacy dictionary is displayed on the left hand side. The developer will map each activity to appropriate actions. For example, the invoke activity in Figure 2 performs read and distribute actions on the data contains in its message. Figure 3(a) shows a partial screenshot of the inspector tool in Step 2 of the inspection process.



Action	Activity
read	InvokeCreditScoreService
distribute	InvokeCreditScoreService

(a) Inspector Tool - Step 2: Associating Activities with Actions



(b) Review of the inspection results: resolving conflicts

Fig. 3. Privacy Inspector Tool

Step 3: Associating Activities with Purposes. In this step, the tool presents the purposes defined in the privacy dictionary. Each action identified in Step 2 is now mapped with its purpose by the process designer. For example, read action may be performed for the homeloan processing purpose and distribute is there for the credit check purpose.

Step 4: Associating PartnerRoles with Users. Finally, the users of the personal data are identified. The partnerRoles listed in the BPEL definition are potential receivers of the personal data. The process designer will map each partnerRole to the user dictionary. For example, the partnerLink in Figure 2 is linked with a parterRole who can be mapped to the user credit checker.

```
<InspectionResults name="homeLoanService">
 <activity name="invokeCreditChecker">
  <bid over="home postal">
    <user>ours</user>
    <action>collect</action>
    <purpose>homeloan processing</purpose>
  </bid>
  <bid over="income">
    <user>ours</user>
    <action>distribute</action>
    <purpose>check credit</purpose>
  </bid>
 ...
 </activity>
 <activity name="invokeRiskAssessor">
 ...
 </activity>
</InspectionResults>
```

(a) The results of inspection for invokeCreditChecker activity

```
<PrivacyTrails name="homeLoanService">
  <activity name="ReceiveHomeLoanRequest">
     :
  </activity>
  <activity name="invokeCreditChecker">
    <trail>
      <right over="home_postal">
       <user>ours</user>
       <action>collect</action>
       <purpose>homeloan processing</purpose>
      </right>
      <obligations>
       <alert_retention activate="180_days"/>
      </obligations>
    </trail>
    <trail>
      <right over="income">
       <user>ours</user>
       <action>distribute</action>
       <purpose>check credit</purpose>
      </right>
      <obligations>
       <write_log activate="immediately"/>
       :
      </obligations>
    </trail>
    :
```

(b) The privacy trails generated for homeLoanService

Fig. 4. Inspection Results / Privacy Trails

Results of the inspection. Figure 4(a) shows the metadata generated as the results of the completed inspection steps. The summary describes each activity in terms of the privacy dictionaries: e.g., invokeCreditChecker's data, users, actions and purposes are identified respectively. We use the term 'bid' (i.e., bid over home postal), because we see this metadata as a request for access to the data.

3.2 Review of Inspection Results

After the metadata has been generated, the privacy inspector automatically compares the bidding content with the rights stated in the privacy policies of the organisation. When the tool finds any conflicts between the content of metadata and the rights in the privacy policies, it highlights the problem which has to be rectified before the BPEL process can be deployed.

As a simple example, Figure 4(a) shows that invokeCreditChecker is bidding for data income to perform distribute action for the purpose of check credit. Let us assume that the combination does not appear in the privacy policies. Then the tool will raise an issue by displaying red coloured lines (e.g., Figure 3(b)). We assume that such problem should be resolved by either changing the privacy policies to reflect the reality of the business process or modifying the implementation of the business process to conform to the policies.

4 Privacy Monitoring Framework

Once the BPEL process has passed the inspection, it is deployed to the execution environment. The main responsibility of BpelPrivacyMonitor during the execution of the deployed process is to monitor its usage of personal data and implement the obligations. For this purpose, we first generate a set of privacy trails which will be attached to the BPEL process for monitoring.

4.1 Generating Privacy Trails for a Business Process

The process of generating privacy trails for a BPEL process is straightforward. The results obtained from the inspection tool are used as the basis. Simply put, each bid over data d in InspectionResults becomes right over d; and the action, purpose and user are appropriately annotated for the right. Then, for each right, obligations are attached. Which obligation should be attached to a right can be determined from the privacy policies.

Figure 4(b) shows the privacy trails generated for the homeLoanService process. It shows, for instance, that the invokeCreditChecker activity will collect home postal data for the purpose of processing homeloan, but it must activate alert retention obligation after 180 days.

4.2 Implementing Obligations

Without referring to the specific implementation platform and architecture we explain how obligations are meant to do.

In the obligation management framework, activities in the BPEL engine execution queue are monitored. As soon as an activity listed in the privacy trails appears, the obligation manager is activated. The manager is passed in enough information to carry out appropriate operations (i.e., the details of the activity and associated obligations).

When the obligation is set to be carried out 'immediately', the obligation manager will run the appropriate class that implements each obligation (e.g., writing log). When the obligation has a time property and if it is greater than 'immediately', the obligation manager registers it as a new timer event with the Time Manager. This sets an alarm with the Time Manager who will trigger the obligation manager when the timer is expired. This will in turn trigger the obligation manager to run the appropriate obligation implementation class. We currently support four types of obligations, however, more obligations can be added as needed. Each obligation implementation class does the following:

write log: The implementation class of this obligation will write the details of the current action to the logs. Although it is possible to associate activation time to this obligation, it is more likely that 'write log' will be carried out immediately. The log includes the <right> section (i.e., data, action, purpose and user) of the privacy trails for the activity and a time-stamp.

send notification: This obligation is implemented to notify the owner of the data of the actions performed on their data. This is likely to be used when the organisation is disclosing the data to a third party (e.g., credit checker in homeloanService). Our current implementation uses email for notification. Similar to logging, the email contains the <right> section of the privacy trails for the activity and a time-stamp.

alert retention: This obligation is designed to alert the system administrator of possible over retention of the personal data collected by a business process. For example, assume that an activity a performs 'collect' action and has (alert retention, 180 days) as an obligation. When activity a occurs, the obligation management framework registers the activity and its obligation alert_retention with the Time Manager. After 180 days from the event, the Time Manager will trigger the obligation manager to run the alert_retention class which sends a warning message to the system administrator.

regular report: This obligation implements dispatching of regular reports to the data subject about the usage. The time property associated indicates how often the reports are sent. Each report contains the summary of the usage over the given period (e.g., last day of each year).

Implementation. For the execution environment, we use ActiveBPEL[4], a well-known open source BPEL engine. To implement our obligation framework, we extend the engine to run in an Aspect-Oriented environment.

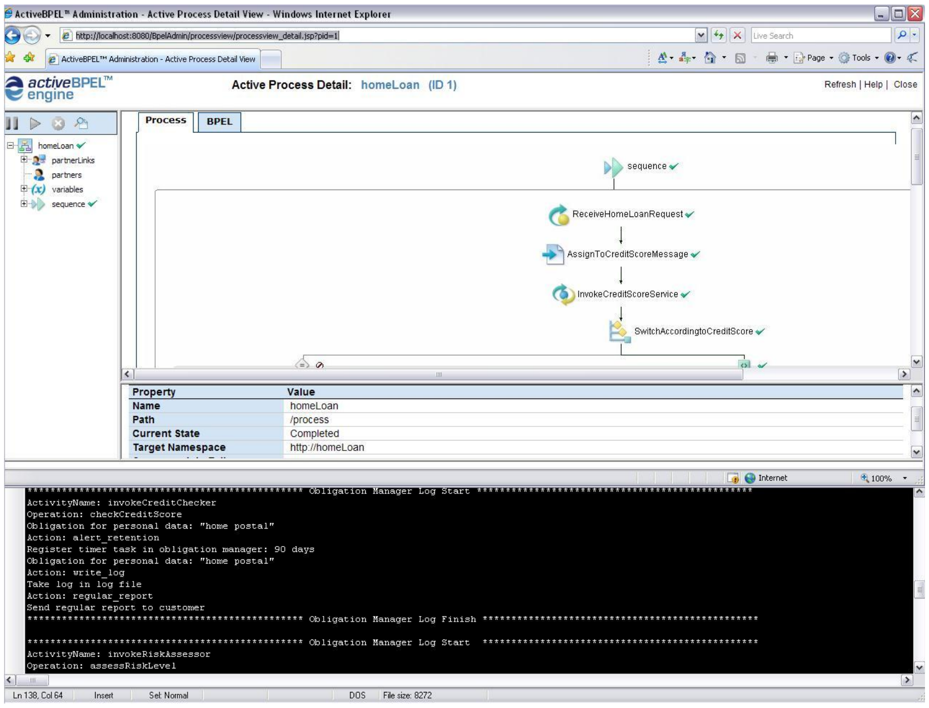
Figure 5 shows the homeloanService process deployed into ActiveBPEL engine which is already extended to run the obligation implementation classes. The console terminal at the bottom shows the results of a few write_log obligations.

5 Related work and Conclusion

We broadly define two related areas: monitoring BPEL processes and privacy enforcement.

Monitoring BPEL Processes. To our best knowledge, there is no work specifically designed to monitor BPEL process for privacy. However, there are some work done in monitoring the execution of BPEL for generic non-functional properties. For example, Baresi et. al. [4] proposed a proxy-based solution for monitoring QoS agreements set between the service client and service provider. A monitoring rule consists of monitoring location, monitoring parameters and monitoring expressions. In Mahbub et. al. [8] presented a framework for monitoring requirements of a composite service. Event calculus are used to describe the requirements that need to be monitored. Requirements can be behavioural properties of a composite service or assumptions about the process and all of its underlying services. It is noted that none of these work directly consider privacy as their primary issue. It will require some ad-hoc privacy data modelling and significant extension for these systems to cater for privacy. An interesting work is proposed by Bettini et. al. [6], which we believe can be complimentary to our future work. The work introduces a way to manage obligations by associating data users with a level of trustworthiness. In case a user fails to fulfil an obligation, a penalty is

[4] www.activebpel.org


ActiveBPEL™ Administration - Active Process Detail View - Windows Internet Explorer
http://localhost:8080/BpelAdmin/processview/processview_detail.jsp?pid=1
activeBPEL™ engine
Active Process Detail: homeLoan (ID 1)
Refresh | Help | Close
Process
BPEL
homeLoan
partnerLinks
partners
variables
sequence
sequence
ReceiveHomeLoanRequest
AssignToCreditScoreMessage
InvokeCreditScoreService
SwitchAccordingtoCreditScore
Internet
100%
Ln 138, Col 64
Insert
Sel: Normal
DOS
File size: 8272

Property	Value
Name	homeLoan
Path	/process
Current State	Completed
Target Namespace	http://homeLoan

```
************************************************ Obligation Manager Log Start ************************************************
ActivityName: invokeCreditChecker
Operation: checkCreditScore
Obligation for personal data: "home postal"
Action: alert_retention
Register timer task in obligation manager: 90 days
Obligation for personal data: "home postal"
Action: write_log
Take log in log file
Action: regular_report
Send regular report to customer
************************************************ Obligation Manager Log Finish ************************************************

************************************************ Obligation Manager Log Start  ************************************************
ActivityName: invokeRiskAssessor
Operation: assessRiskLevel
```

Fig. 5. Screenshot of ActiveBPEL engine and write_log obligations

applied (e.g., decreasing of trustworthiness). We believe, for example, logs generated by BpelPrivacyMonitor can be used to automatically detect the violation of obligations and apply the penalties.

Privacy Policy Enforcement. Enforcing privacy policies on the business applications has been largely approached from traditional access control view point. For example, Agrawal et. al. have introduced the concept of Hippocratic databases in [3]. The core idea is to exploit an SQL rewriting technique to incorporate policy evaluation into a database query to ensure that data is only disclosed to appropriate people. The implementation assumes relational database systems and it requires analysis of each SQL query for re-writing. Another database-centric approach is presented in [9], in which the author focuses on the management and enforcement of privacy obligations in a closed enterprise environment. The approach does not take purpose of access, the core concept of privacy, into consideration.

The drawback of these approaches is that they focus on the database level (relational databases in particular). We believe that a higher-level view (i.e., business process and orchestration) consideration is required to cater for the privacy concerns that may span across different systems and different types of data repositories.

Schunter et. al [10] has proposed the Privacy Injector system which leverages the Aspect-Oriented programming paradigm to implement privacy enforcement. The system associates privacy metadata to each piece of personal data that is collected. All data-manipulation operations should work within the boundary of what the metadata dictates. All usage and disclosure of particular personal data are checked and the necessary actions are performed. One thing to note is the system performance aspect. Checking every action against the privacy metadata will undoubtfully be inefficient. In our approach, we move some of the checking to the design phase through the inspection process to avoid checking every action at runtime. Also, our approach proposes a generic obligation management whereas the privacy injector system mainly focuses on management of data disclosure.

To conclude, we introduced BpelPrivacyMonitor, a privacy inspection and monitoring framework for automated business processes. In particular, we considered processes implemented in BPEL. BpelPrivacyMonitor provides two functions: (i) the process designers can look over the privacy aspects of an automated business process and resolve potential problems before deployment, and (ii) the obligation management technique ensures when certain actions performed on personal data, appropriate obligations are carried out. We are currently working on how to exploit the logs generated by our framework. The aim is to assist the privacy audit process by using the knowledge obtained from the analysis of the process logs with respect to privacy.

References

1. BPEL 1.1. www.ibm.com/developerworks/library/specification/ws-bpel.
2. DAML services: Security and privacy. www.daml.org/services/owl-s/security.html.
3. R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic databases. In *VLDB*, pages 143–154, 2002.
4. L. Baresi and S. Guinea. Towards dynamic monitoring of ws-bpel processes. In *ICSOC*, pages 269–282, 2005.
5. S. Benbernou, H. Meziane, Y. Li, and M-S. Hacid. Privacy-agreement for web services. In *SCC*, 2007. to appear.
6. C. Bettini, S. Jajodia, X. Wang, and D. Wijesekera. Obligation monitoring in policy management. In *POLICY*, page 2, 2002.
7. Y. Li, H-Y. Paik, B. Benatallah, and S. Benbernou. Formal consistency verification between bpel process and privacy policy. In *Privacy Security and Trust*, pages 212–223, 2006.
8. K. Mahbub and G. Spanoudakis. A framework for requirents monitoring of service based systems. In *ICSOC*, pages 84–93, 2004.
9. M. C. Mont. Towards scalable management of privacy obligations in enterprises. In *TrustBus*, pages 1–10, 2006.
10. M. Schunter and C. V. Berghe. Privacy injector: Automated privacy enforcement through aspects. In *Workshop on Privacy Enhancing Technologies*, pages 99–117, 2006.