

Security Views for Outsourced Business Processes*

Azzedine Benameur
SAP Research, Security&Trust
Avenue du Docteur Maurice
Donat 805
Mougins, France
azzedine.benameur@sap.com

Fabio Massacci
The University of Trento
via Sommarive 14
Povo, Italy
fabio.massacci@unitn.it

Nataliya Rassadko
The University of Trento
via Sommarive 14
Povo, Italy
nataliya.rassadko@unitn.it

ABSTRACT

The workflow of a Virtual Organization is often divided into fragments that are run by different entities having different clearance level or accessibility permissions. Therefore, an important issue is a decomposition of the overall business process into workflow views that can be outsourced to the side of the corresponding contractors. In this paper, we introduce the notion of *business process security view* and present an algorithm for the automatic derivation of such views from a security specification that may express conditional accessibility based on the actual data flowing across business process. Our solution borrows the idea of virtual views from relational database views. We also discuss an architecture and an implementation for workflow view synchronization.

Categories and Subject Descriptors

D.2.11 [Software Engineering]; D.4.6 [Security and Protection]

General Terms

Security Design

Keywords

Business Process, Workflow, Security View, Access Control, Electronic commerce

1. INTRODUCTION

The last decade has seen a major reshaping of the way business is conducted over the web:

- *service-oriented architectures and business (SOA) process management platforms* [6] are emerging as the architectures and technologies of choice for structuring and integrating applications within and across enterprises.

*This research has been partially supported by the the projects EU-FP7-IP-MASTER , EU-FP6-IP-SERENITY, and EU-FP6-IP-SENSORIA.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SWS'08, October 31, 2008, Fairfax, Virginia, USA.

Copyright 2008 ACM 978-1-60558-292-4/08/10 ...\$5.00.

- companies and institutions are often *outsourcing fragments of their business processes*. Outsourcing is sometimes iterated, so that the service provider itself outsources some functions to the third parties [13].

From the business perspective, it is seldom the case that the (outsourced) process can be “zipped” into the invocation of a single web-service. Rather whole workflows must be orchestrated and run in a decentralized manner. This happens in business-to-business cooperations [12, 16, 10] in mobile environment [3, 9], in grid systems [29], in peer-to-peer networks [5, 20, 30], etc.

From a security perspective, outsourcing has a profound impact on the trust models, security policies, security procedures, and security infrastructures that companies need to develop and maintain [19, 18]. One of the major changes over traditional models of access control for workflows [4, 17, 26] is that we can no longer assume that there is a single entity in charge of the enforcements of access control rules.

Let's consider the “classical” loan origination workflow from A. Schaad's papers on separation of duties [26]. It is necessary to enforce separation of duties (SoD) between the front office (responsible for client loan request reception) and the back office (approving the loan request or not). Enter outsourcing and we end up with not only two physically different offices but also managed by different companies and with different ownership and trust relations. The central bank holding no longer wants the entire workflow to be visible to the various subsidiaries.

Of course it is possible to manually break down the overall company workflows into separate sub-workflows but this would not scale up to dynamic Virtual Organizations [29, 14].

1.1 The Contribution of this paper

In order to solve this problem, we introduce the notion of *business process security views* (Business Views for short). Business Views are similar to database views in the sense that they reveal to actors all and only accessible information on tasks to be executed. In the same fashion of database views, when the underlying database changes the view is updated and the users see a different process.

With an appropriate definition of the access control rules on the global company workflow, we can construct the workflow view that the outsourced contractor (corresponding to the schema of the database view) should access and ship it to him for execution with a local Workflow Management System (WFMS) at the contractor's side. The mother company can simply run a *stateful* mediator system [27], where synchronization of workflow views is done by one central participant that routes information among local WFMSs (if necessary). This central holding may log interactions, perform modification of data, assign scheduled tasks to partners, etc.

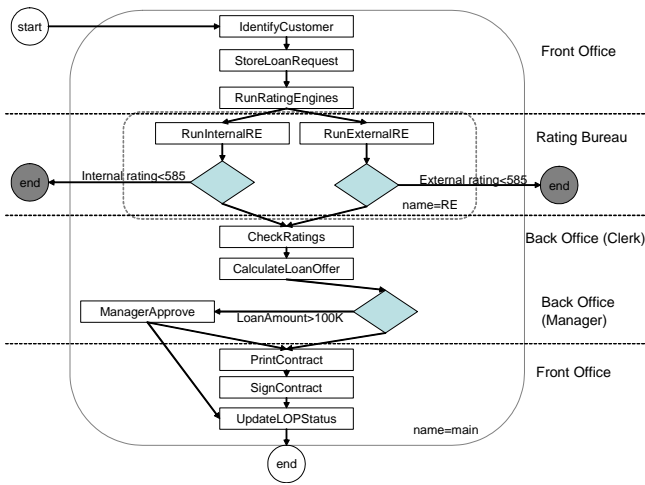


Figure 1: General Loan Origination Process, BPG

In a nutshell, the contributions of the paper are the following:

- Formalization of business process by regular expressions called *flow control rules*.
- Security specification language that can express also conditional rules, where conditional accessibility depends on data flowing across business process.
- Algorithm for business process view construction according to an authorization specification.
- Architecture for business process outsourcing.
- Architecture implementation.

The rest of the paper is organized as follows. In Sec. 2, we present our motivating example: the Loan Origination Process (LOP). Next, a formalization of business process is defined in Sec. 4. The notion of workflow views is given in Sec. 2.1. Then we introduce the language for authorization specification in Sec. 4, and workflow construction according to the authorization specification in Sec. 5. Next, synchronization protocol among decentralized views is explained in Sec. 5.1. Implementation issues of the proposed model are discussed in Sec. 6. In Sec. 7, we present a short survey of related work. Finally, we conclude the paper in Sec. 8.

2. MOTIVATING EXAMPLE

The starting point for our running example will be the “classical” loan origination process (LOP) from SAP mostly popularized by A. Schaad et al. [26].

LOP describes a customer wanting to buy a product whose offering depends on the outcome of a number of actors. Several external and internal ratings need to be obtained in order to assess the credit worthiness of the customer (internal rating, Credit Bureau). Once the credit worthiness of the customer has been positively established, the “bank” selects a bundle product and submits it to the customer. If the customer is satisfied by the proposed product, both parties come to an agreement and sign a contract. We used quotation marks because what the customer naively consider a single entity is in reality a virtual organization.

Figure 1 illustrates the different steps of the scenario, with the actors and the roles involved. Namely, the workflow sequence (enclosed in a rectangle with a solid border) starts after the customer

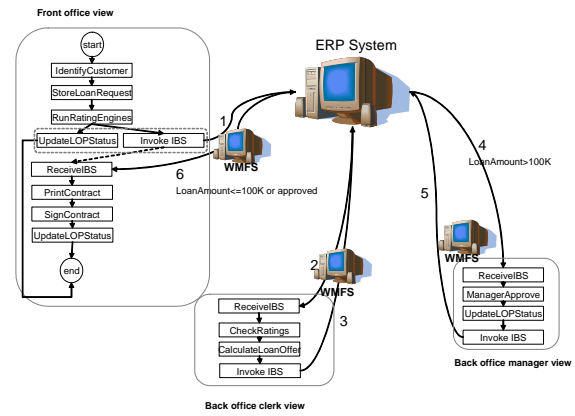


Figure 2: BPEL view outsourcing

makes a loan request. Then, the front office checks identity of the client. After that, two parallel rating engines run (depicted as a rectangle with a dotted board) and customer rating is calculated: the credit worthiness of the customer is checked by querying an external rating engine (e.g., Credit Bureau), an internal check about the credit worthiness of the customer has to be done as well. In the case of a low rating, the rating engines terminate with error (depicted as grey end point). Otherwise, the back office clerk calculates the price of the loan depending on the rating of the client. If the loan exceeds 100K, the loan offer should be approved by a back office manager. Finally, the bank and the customer come to an agreement and sign the contract at the side of the front office.

Basically, the back office and the front office may be different organizations dislocated geographically and organizationally. Therefore, we need to develop a mechanism that calculates, outsources and synchronizes the corresponding part of the business process.

Critical tasks as customer identification and loan offer calculation are separated between front and back offices. For example, the back office should not be able to identify customers. On the other hand, front office, even knowing the requested loan amount, should not influence the loan offer. This separation might be due to classical SoD constraints or different business constraints (e.g., the back offices offering the same functionalities to different “banks”).

At this point, we would like to introduce our security model that associates a different business security views to different roles. Classical roles in the loan originating process (pre-processing clerk, post-processing clerk or manager) are replaced in our scenario by the contractors. The resulting model in our scenario is thus depicted in Fig.2.

Example 2.1: Front office receives the customer, identifies him, stores loan request, and sends request for starting rating engines to the internal banking system (IBS). IBS runs internal rating engine and external rating engine to calculate a score of the customer. IBS assigns the processing of calculated ratings to the back office. As Fig. 2 shows, the back office invokes IBS after loan offer calculation. Then, IBS will assign the task of LOP finalization that includes printing and signing the contract at the site of the front office. From the viewpoint of the front office clerk, the contract printing Web-service is *asynchronous* (dotted line in Fig. 2), since its invocation is postponed by the back office workflow. □

2.1 Business Process Views

Taking into account that IBS from the motivating example not only specifies actor collaboration but also may run its own tasks,

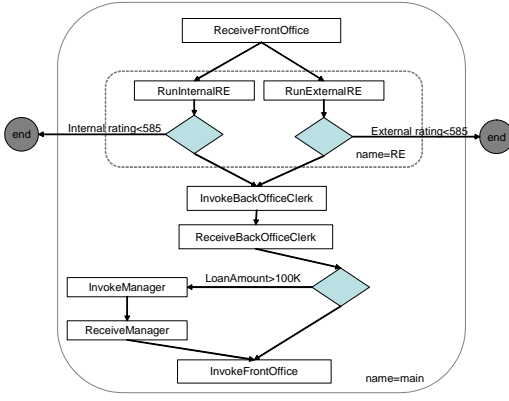


Figure 3: Bank view of workflow

e.g., *RunInternalRE*, we infer that IBS is a workflow coordinator and has its own view of the business process. An example of the bank view is shown in Fig. 3.

In the sequel, we assume that for the assignments of roles to contractors and their validation one can use any existing authorization mechanisms in order to concentrate on the novel aspects of Business Process Views (BP Views).

3. BUSINESS PROCESS MODELS

As we have seen in the motivating example, a business process may be modelled as graph, where nodes are atomic tasks and edges represent a flow among tasks. A basis for orchestration specification of business processes is a sequential/parallel composition of tasks and conditional statements that are expressed by the following BPEL constructs:

1. *sequence*, i.e., set of tasks are organized into a precise sequence that should be respected;
2. *flow*, i.e., set of tasks may be run concurrently;
3. *if*, i.e., only one task from a set of parallel tasks is run in the current sequence of tasks if some condition holds.

We now present a formalism for representation of business process as a graph¹.

Definition 3.1: *Business Process Graph (BPG)* is a tuple $\langle start, end, \Theta, C, P, \chi \rangle$, where

- *start* is a root of the graph called *starting point of the process*.
- *end* is a set of graph nodes called *ending points of the process*.
- Θ is a set of nodes called *tasks*². Each task we denote as τ .
- C is a set of conditions.
- P is a set of graph nodes called *control flow relation* that for each input τ defines a set of outputs τ' and is denoted as $\tau \rightarrow P(\tau) = \alpha$, where α is a regular expression, defined as follows:

$$\alpha := end \mid \tau \mid \tau + \tau \mid if(c, \tau_c, \tau_{else}?)$$

¹For our implementation business processes are expressed using Business Process Execution Language [6]. Algorithm for deriving BPEL from BPG is reported in appendix A.

²Neither starting point nor ending points are tasks

- $\chi(\tau)$ is a function that returns an XPath expression³ representing a nesting of flow structures called *containers* in which τ is included: *flow*, *sequence*, *if*. The expression $\chi(\tau)/name$ receives the label of container if it is not null.

Due to the nature of ending points we assume that $P(end) = \emptyset$. \square

In other words, the control flow relation for a task τ may be *end*, or another task τ' executed sequentially, or a set of tasks executed in parallel, or conditional flow where c is a condition, τ_c is a task executed sequentially if $c \in C$ holds, τ_{else} is an optional task (marked with symbol “?”) executed sequentially if condition c does not hold.

Example 3.1: According to Def. 3.1, the formal representation of the workflow of our motivating example is the following. BPG G is a tuple $\langle start, end, \Theta, C, P, \chi \rangle$, where

$\Theta = \{CustomerIdentification, CheckRating, CalculateLoanOffer, \dots\}$;

Control flow rules are

```

P(IdentifyCustomer) = (StoreLoanRequest)
P(StoreLoanRequest) = (RunRatingEngines)
P(RunRatingEngines) = (RunInternalRE+RunExternalRE)
P(RunInternalRE) =
if(InternalRating<585, end, CheckRating)
P(CheckRating) = (CalculateLoanOffer)
P(CalculateLoanOffer) = if(LoanSmount>100,
ManagerApprove, PrintContract)
P(ManagerApprove) = (PrintContract+UpdateLOPStatus)
P(PrintContract) = (UpdateLOPStatus)
P(UpdateLOPStatus) = (end)

```

Examples of *chi*-function: $\chi(CheckRating)=sequence$ because *CheckRating* is in the first sequence, $\chi(CheckRating)/name=main$;
 $\chi(RunInternalRE)=sequence/flow$ because *RunInternalRE* is in parallel flow which is nested into a sequential flow,
 $\chi(RunInternalRE)/name=RE$. \square

4. ACCESS CONTROL ON BUSINESS PROCESS

We can now introduce our access control model for business processes.

Definition 4.1: An *access specification* is a pair (G, ann) , where G is a BPG, ann is a partial mapping such that for a control flow relation $\tau \rightarrow P(\tau)$, for element $\tau' \in P(\tau)$:

$$ann(\tau, \tau') := Y \mid N \mid Q[q]$$

where Y and N denote that business process task τ' is accessible or inaccessible respectively after the business process task τ being executed. The qualifier $Q[q]$ is a conditional accessibility, where q is an expression depending on data transmitted across the workflow. We assume that $ann_{data}(start)=Y$ by default. \square

Qualifier $Q[q]$ is used to formulate accessibility conditions depending on data, e.g., values of input and output parameters for orchestrated Web-services. The qualifier means that the task is permitted if condition holds. Otherwise, the task is forbidden.

If $ann(\tau, \tau')$ is not explicitly defined, τ' may *inherit* an annotation from τ or may gain an annotation according to some *default* policy. On the other hand, if $ann(\tau, \tau')$ is explicitly defined, it cannot be overridden with a propagated annotation. In the current paper, we consider the case when annotation labels are propagated

³BPEL is an XML dialect. Therefore, navigation in BPEL may be done via XPath.

top-down, i.e., from the start point to the end points. Our algorithms can be easily extended to bottom-up propagation as well.

Since we put annotations on BPG edges, the idea behind our algorithm is to “push” security labels from edges to tasks. The labeled task, in its turn, may propagate its annotation to the tasks that follows it via corresponding BPG edges that were initially unlabelled. In this way the security model guarantees that the *most specific rules takes precedence* principle is followed.

Example 4.1: For the front office, the annotation contains the following labels:

```
ann(RunRatingEngines, RunInternalRI) = N
ann(RunRatingEngines, RunExternalRI) = N
ann(ManagerApprove, PrintContract) =
Q[ManagerApprove = true]
ann(if(LoanMount < 100), PrintContract) = Y
```

The back office clerk has other labeling:

```
ann(start, IdentifyCustomer) = N
ann(RunInternalRI, CheckRating) = Y
ann(RunExternalRI, CheckRating) = Y
ann(CalculateLoanOffer, ManagerApprove) = N
```

The qualifier means that the workflow view for the post-processing clerk will continue if the manager approves the loan with an amount superior to 100K. However, in any case, the post-processing clerk should report a status of the LOP. \square

5. BUSINESS PROCESS VIEW CONSTRUCTION

We now show how to construct a BPG security view, given a BPG structure and an authorization specification. The derivation of BPG view has two parts:

1. computing a fully annotated BPG such that Y/N labels are assigned to *every* element type, and
2. restructuring the fully annotated BPG so that N-labelled elements are deleted and their permitted children are attached to their nearest permitted ancestors. In this manner, the BPG view shows all, and only, the accessible data.

In parallel with BPG view construction, we maintain an additional σ -function that will be used by the centralized mediator to translate the current flow from the centralized entity to the contractors and viceversa.

Example 5.1: In Example 4.1, there is an annotation $\text{ann}(\text{start}, \text{IdentifyCustomer})=Y$. It means that from *IdentifyCustomer* we should propagate Y to the following task, i.e., *StoreLoanRequest*. \square

First of all, we have to eliminate qualifiers. As we said above, qualifiers mean different instantiation of a business process. Therefore, we duplicate the task, which accessibility is defined by a qualifier, to a permitted task and a forbidden task. We create them as a parallel flow. At the end, the flows starting with the forbidden duplicate will be deleted and if it is finalized by a permitted task, this task should be started in parallel with a permitted flow of a duplicated task as an asynchronous Web-service.

As Fig. 1 shows, BPG is not a tree. Therefore, it may happen that different annotations may arrive to the same task. It means that in some workflow instances, a certain BPEL portion will be executable, while in others not. Thus, we have to duplicate such a task as well.

Example 5.2: The front office should execute tasks *PrintContract* and *SignContract* if the approval decision of the manager is `true`.

Algorithm ANNOTATE BUSINESS PROCESS GRAPH

Input: A authorization specification (G, ann)

Output: Fully annotated BPG G'

```
1: Initialize  $G' := G$  where  $\text{ann}$  is defined on  $G'$  as on  $G$ ;
2: for all control flow relations  $\tau \rightarrow P(\tau)$  in  $G'$  and all  $\tau' \in P(\tau)$  do
3:   initialize  $\sigma(\tau, \tau') := \tau'$ 
4: end for
5: for all  $\tau \rightarrow P(\tau)$  and all  $\tau' \in P(\tau)$  with  $\text{ann}(\tau, \tau') = Q[q]$  do
6:   add to  $G'$  a new element type  $\tau'_N$  and a control flow rule  $\tau'_N \rightarrow P(\tau')$ 
7:   in  $P(\tau)$  substitute  $\tau'$  with  $\tau' + \tau'_N$ 
8:   set  $\sigma(\tau, \tau') := \tau'[q]$ ;  $\sigma(\tau, \tau'_N) := \tau'[\neg q]$ ;
9:   set  $\text{ann}_{\text{data}}(\tau') = Y$  and  $\text{ann}_{\text{data}}(\tau'_N) = N$ ;
10:  for all element types  $\tau''$  occurring in  $P(\tau')$  do
11:    set  $\sigma(\tau'_N, \tau'') := \sigma(\tau', \tau'')$ ;
12:    set  $\text{ann}(\tau'_N, \tau'') := \text{ann}(\tau', \tau'')$ ;
13:  end for
14: end for
15: while  $\text{ann}_{\text{data}}(\tau')$  of some task  $\tau'$  is undefined do
16:  if all preceding tasks  $\tau$  of  $\tau'$  have defined  $\text{ann}(\tau, \tau')$  then
17:    if all  $\text{ann}_{\text{data}}(\tau) = Y$  then
18:      set  $\text{ann}_{\text{data}}(\tau') := Y$ ;
19:    else if all  $\text{ann}_{\text{data}}(\tau) = N$  then
20:      set  $\text{ann}_{\text{data}}(\tau') := N$ ;
21:    else
22:      add to  $G'$  a new task  $\tau'_N$  and a control flow relation  $\tau'_N \rightarrow P(\tau')$ 
23:      set  $\sigma(\tau, \tau'_N) := \tau'$ ;
24:      set  $\text{ann}_{\text{data}}(\tau') = Y$ ,  $\text{ann}_{\text{data}}(\tau'_{no}) = N$ ;
25:      for all element types  $\tau''$  occurring in  $P(\tau')$  do
26:        set  $\sigma(\tau'_N, \tau'') := \sigma(\tau', \tau'')$ ;
27:        set  $\text{ann}(\tau'_N, \tau'') := \text{ann}(\tau', \tau'')$ ;
28:      end for
29:      for all preceding tasks  $\tau$  of  $\tau'$  do
30:        if  $\text{ann}_{\text{data}}(\tau) = N$  then
31:          replace  $\tau'$  with  $\tau'_N$  in  $P(\tau)$ 
32:        end if
33:      end for
34:    end if
35:  end if
36: end while
```

Figure 4: Algorithm ANNOTATE BUSINESS PROCESS GRAPH

It should also finalize workflow by reporting a status of LOP, e.g., “contract is signed”. Otherwise, the contract cannot be either printed or signed. However, LOP status should be updated, e.g., “loan offer calculated”. The latter should be done in parallel with invocation of IBS. \square

An overall algorithm for BPG annotation propagation is shown in Fig. 4. Lines 2-4 show an initialization of σ -function. In lines 5-14, we eliminate qualifiers that express access condition on data. Basically, at this step, we emulate two possible outcomes of *if* construct. Namely we split the task following *if* into create Y-labeled and N-labeled copies. And the propagation continues. Having eliminated all qualifiers, we simply propagate Y and N labels in lines 15-21 If both positive and negative labels arrive to the same node, we repeat the same splitting operation to resolve conflict (lines 21-34).

Having fully annotated BPG, we have to eliminate forbidden

Algorithm BUILD VIEW

Input: Fully annotated BPG G
Output: A security view (G', σ)

```
1: for all tasks  $\tau$  with  $\text{ann}_{\text{data}}(\tau) = N$  do
2:   for all control flow rules  $\tau \rightarrow P(\tau)$  do
3:     if  $\tau'$  occurs in  $P(\tau)$  then
4:       for all  $\tau''$  that occurs in  $P(\tau')$  do
5:         set  $\sigma(\tau, \tau'') := \sigma(\tau, \tau')/\sigma(\tau', \tau'') \cup \sigma(\tau, \tau'')$ 
6:       end for
7:       if  $P(\tau') \neq \emptyset$  then
8:         replace  $\tau'$  by  $\text{InvokeIBS}(\text{user}, \tau, \text{input}(\tau'))$  in  $P(\tau)$ ;
9:         create  $P(\text{InvokeIBS}) = \text{ReceiveIBS}(\text{output}(\tau'))$ ;
10:        create  $P(\text{ReceiveIBS}) = P(\tau')$ ;
11:        delete  $\tau'$ ;
12:       else
13:         replace  $\tau'$  by  $\text{end}$ ;
14:       end if
15:       if  $\tau$  is  $\text{ReceiveIBS}'$  then
16:         delete  $\text{ReceiveIBS}'$ ;
17:         delete  $\text{InvokeIBS}$ ;
18:       end if
19:     end if
20:   end for
21: end for
```

Figure 5: Algorithm BUILD VIEW

tasks and insert calls of IBS and task assignments from IBS. During elimination of forbidden tasks, we update the σ -function in such a way that it can be used to reconstruct the original workflow. For the invocation of IBS and reception of responses from it, we introduce virtual tasks *invokeIBS* and *receiveIBS* respectively. The first one accepts parameters of those deleted tasks that follow immediately after the last permitted tasks. For example, *RunInternalRE* is a forbidden task for pre-processing clerk and follows immediately after *RunRatingEngines*. The second virtual task, *receiveIBS* issues the values for parameters of those permitted tasks that follow immediately after the last forbidden tasks. For example, *CheckRating* will be called after *receiveIBS* that issues values calculated by *RunInternalRE* and *RunExternalRE*, which are the last forbidden tasks for pre-processing clerk. The formalization of algorithm BUILD VIEW is shown in Fig. 5.

5.1 Rewriting of Virtual Tasks

In the previous section, we introduced virtual tasks *receiveIBS* and *invokeIBS*. Their implementation may be done in two ways. The first one is a physical creation of Web-services corresponding to those tasks. In such a case, we should automatically generate, e.g., Java-code for these Web-services and a script that compiles the generated code, creates WSDL-description and deploys Web-services on an application server. This approach is similar to view materialization in relational databases and has similar drawbacks. Namely, we have the problem of integrity maintenance among created Web-services when access policy changes or workflow is modified: we have to recycle previously created services and delete obsolete ones. This requires a sophisticated mechanism for service usage tracking. The infeasibility of this solution is self-evident.

As a better alternative, we developed a technique of invocation rewriting which is similar to query rewriting and virtual views in relational databases. The idea is to rewrite calls to artificial Web-services into calls to real Web-services. For this purpose, in the front of every local WFMS, there should be a special handler or gateway. Central ERP handler (i.e., handler of IBS) is smarter in the sense that it has information about location of other handlers and accessibility of a corresponding agents participating in business process. Each agent handler knows only a location of the IBS handler.

The protocol of synchronization is twofold: from an agent to IBS and from IBS to an agent. Below we define it in a semi-algorithmic way:

Invocation of IBS by an agent. Assume that the agent executed a part of its workflow and received some data *result* that should be passed to IBS.

1. *invokeIBS* should be done with two parameters: *result* and *last_executed*, which is the last executed task by the agent.
2. Agent's handler rewrites this call to $\text{Invoke}(\text{IBS}, \text{Receive}(\text{Agent}, \text{result}), \text{last_executed})$.
3. IBS handler intercepts this call and rewrites it into $\text{ReceiveAgent}(\text{result})$.
4. IBS assigns *result* to $P(\text{last_executed})$ which is a set of tasks in the original workflow that should follow *last_executed*.
5. IBS starts to execute τ and then the rest of the workflow view related to its WFMS.

Invocation of an actor by IBS. Suppose that IBS

should pass *result* of the preceding workflow execution to some agent. We denote a set of the last executed workflow tasks as *last_executed* (either received from some agent, or executed by IBS itself).

1. IBS calls $\text{invokeAgent}(\text{result})$.
2. IBS handler rewrites it to $\text{Invoke}(\text{Agent}, \text{receiveIBS}(\text{result}), P(\text{last_executed}))$, where $\tau \in P(\text{last_executed})$ is a task that should follow *last_executed* in the workflow related to the agent. The exact identification of τ is done according to σ -function calculated according to the agent's permissions.
3. Agent handler rewrites it into $\text{receiveIBS}(\text{result})$ at the point of the agent's workflow before tasks τ .
4. WFMS of agent assigns *result* to tasks τ and starts to execute them and then the rest of the workflow view related to agents WFMS.

The protocol of workflow view synchronization demonstrates a modular nature of our proposal. Namely, in the current example, IBS with the back and front offices as a whole may be the third party office in other workflow with a different ERP.

6. IMPLEMENTATION ISSUES

In Sec. 2, we presented a general LOP. In this section, we detail how we implement this scenario, our architecture for the BP Views implementation, and finally, we show our solution on rewriting virtual tasks into the real ones that can be integrated seamlessly into existing authorization frameworks.

To implement this scenario we assign several services to the different entities: front office, back office, back office of the manager. So far, we have implemented a simplified version of a workflow having excluded a manager approval task.

- The front office is in charge of identifying the customer, storing the loan request and finalizing the contract. These functions are performed respectively by the following Web-services: WS Authentication, WS Store Loan and WS Loan Calculation.

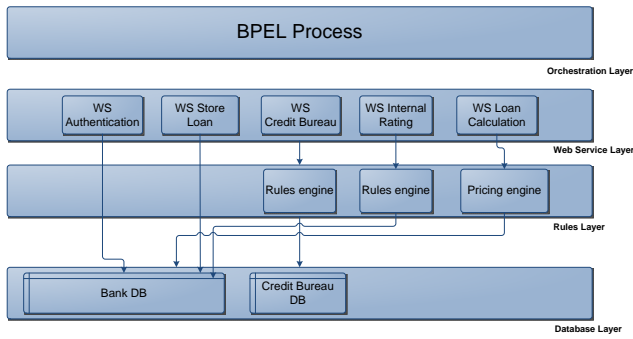


Figure 6: Loan Origination Process Architecture with Web Services

- The IBS is invoked by a call to an artificial Web-service (InvokeIBS) that is rewritten as a call to a real Web-service, as presented in Sec. 5.1. The services representing the IBS view are WS Internal Rating and WS External Rating.
- The back office view of the workflow is modeled as an interface of the WS Calculation Offer.

The logical architecture of the LOP implementation is depicted in figure 6. Four layers are needed to fully decouple the business logic from the functional implementation. A database layer makes a clear separation between the database of the bank computing the internal rating, and the credit bureau database providing an external assessment. The Rules layer enables decoupling of the business rules from the code in such a way that if regulations (or practices) of the partner evolve, the only layer to update is the rules layer. The Web-service layer is complemented by the orchestration layer that contains the BPEL specification of the process that is interpreted by a WFMS. The latter describes how web services works together and on which interface.

The BPEL view architecture involves the back office and the front office that can be dislocated geographically. We therefore assume that each business partner has its own XML security gateway (linked by two solid lines in Fig. 7), where we can place our rewriting module acting according to protocol presented in Sec. 5.1. The overall architecture is depicted in Fig. 7.

Each agent's gateway knows the IBS location (dotted lines), but only the IBS gateway knows the location of the other business partners. Architecture for rewriting can be designed, from our view point, in two ways. The first one enables integration with existing framework such as PERMIS [7], the second allows a integration within the Web-service environment. Below, we explain both of them.

6.1 XML Security Gateway.

In service oriented architecture, communication is performed through XML-based messages called SOAP messages. This communication can be maintained by an XML security gateway. Namely, in the context of SOA, XML gateway is a hosting of Web-service interfaces. These interfaces define endpoints of Web-services. The XML security gateway is also a specialized service that performs security policy enforcement and rewriting of virtual tasks. Therefore, the gateway contains Policy Enforcement Point (PEP) in charge to carry out the policy of the agent or ERP (e.g., accept only signed message, no messages from a certain IP, etc), and a rewriting that rewrites incoming and outgoing calls according to the protocol presented in Sec. 5.1. Figure 8 illustrates the overall principle.

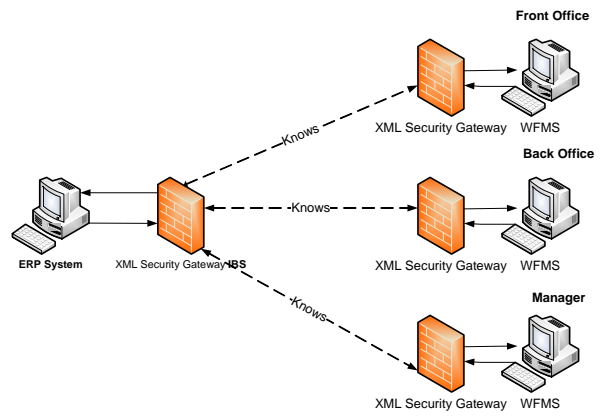


Figure 7: Business Process View Architecture

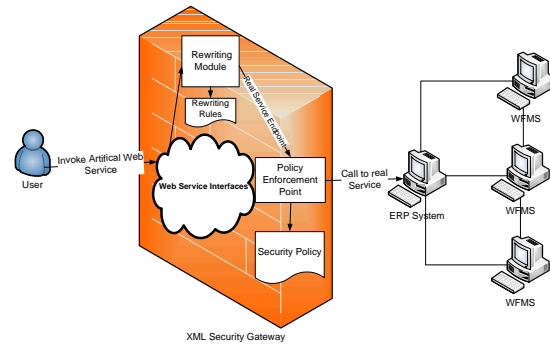


Figure 8: XML Security Gateway

6.2 Web Service Handler.

Our actual Web-service implementation is based on the basis the Apache Axis 2 Web-service engine [2]. This engine provides a particular execution model depicted in Figure 9. It is based on a pipe of sequential actions performed by handlers, which can be cascaded. Handlers can be placed at both the client and the server side (respectively the left part and the right part of Figure 9). It allows to interact with incoming and outgoing messages before they reach out Transport sender which will form the SOAP message and send it. This is the place where we implement our Rewriting module. Thus, on the server side where the message will be rewritten before reaching the Web-service business logic on the server side (right part of figure 9). On each incoming message, it checks whether the target endpoint is an artificial Web-service and then rewrites it.

7. RELATED WORK

Security models and mechanisms for workflows have mostly focussed on the classical issue of separation of duties [25]. In [26], Schaad et al. focused their work on a banking scenario where the "four-eye principle" is important to avoid frauds. They presented a comprehensive taxonomy of SOD, for static and dynamic aspect. The latter is decomposed as follows: Simple Dynamic SOD, Object-Based SOD, History-Based SOD. Finally they applied a model-checking approach to analyze their case study. Recently SOD has been studied within Workflow Management Systems [4], the authors proposed a solution based on an offline syntactic analysis of all the role/user/permission assignment possible that satisfies the policy, however the solution is limited to centralized workflows. In [8], the authors studied SOD in workflows, and considered the

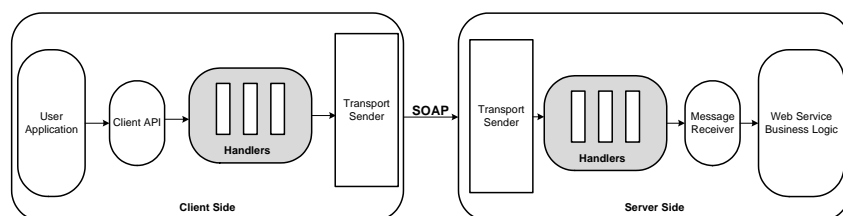


Figure 9: Apache Axis2 Execution Model

fact that a workflow can have several instances and that a user does not necessarily endorse the same role in each instance. They provided a solution to express Multiple-Session SOD policy (MSOD) and to enforce them in a RBAC infrastructure and implemented it into the PERMIS framework [7]. The limitation of MSOD is that it requires a full RBAC infrastructure, moreover the Policy Decision Point (PDP) must know the name or ID of the user who activate a role in order to link the access control session with the user name.

In the business domain, the CrossFlow project [14] aimed at the support of workflows in dynamically created virtual organizations. The architecture is built on proxy gateways that protect or encapsulate the provider-consumer core services of the organizations. In [15], Jablonski et al. presented Agent Enhanced Workflow based on planning using a goal model. ADEPT [23] is a formal foundation for the support of dynamic changes of running workflow instances. Client-server paradigm for distributed workflow is supported in MENTOR [22] and Exotica [21]. A centrally coordinated approach executing cross-organizational workflows resulted in WISE [1]. Wide area workflow management is covered in [24]. Other decentralized WFMSs similar to peer-to-peer architecture were published in [5], [9], [20], [30]. All the above mentioned papers are concentrated on workflow composition, distributed fault handlers, rather than on security properties enforcement and workflow decomposition into views.

Some attempts to introduce business process views were done in the following range of papers. Cross-organization workflow cooperation was proposed in [11], [16]. Later, in [28], business process view was generalized so that to be public and private at the same time. Basically, private view is a more detailed public view that hides sensitive tasks. Public views are considered as partitions of a sketch of the overall business process. Shultz et al. in [27] investigates stateless and statefull mediated communication for cross-organizational workflow architecture, where organizational workflow views are obtained by partitioning the original workflow. The partitioning itself should be done manually. An intuition of a formalism for partitioning rules was given in [3] that is concentrated on mobile networks and federations of business processes. However, an enforcement of these rules is also performed manually.

8. CONCLUSION

To the best of our knowledge, this paper is the first to show the method of *automatic* construction of security views for business processes. Our proposal adopts the idea of virtual views of database. Namely, from a security specification, we automatically calculate a business process view that is then outsourced to a corresponding actor.

We presented a flexible architecture, where each actor application runs their own WFMS in the same fashion as decentralized workflows. One may think that this might be a sort of limitation. On the other hand, our approach is adaptable to pervasive/decentralized workflow with some modifications.

As the future work, we would like to consider a centralized management of workflow, where agents run their portions of business process directly on WFMS of ERP. We would like to develop an access control model to streaming data which is an instantiated business process. After that, it will be interesting to investigate if an instantiated view of business process will be equivalent to an instantiated business process of the view calculated according to the current proposal. We are also interested in extensions of our authorization specification with temporal constraints for enforcement of a dynamic separation of duties and usage control for SOA.

9. REFERENCES

- [1] G. Alonso, U. Fiedler, C. Hagen, A. Lazcano, H. Schuldt, and N. Weiler. Wise: Business to business e-commerce. In *Proceedings of the Ninth International Workshop on Research Issues on Data Engineering: Information Technology for Virtual Enterprises (RIDE'99)*, page 132, Washington, DC, USA, 1999. IEEE Computer Society.
- [2] Apache axis2 web service engine. http://ws.apache.org/axis2/modules/rampart/1_2/security-module.html, 2007.
- [3] L. Baresi, A. Maurino, and S. Modafferi. Towards distributed BPEL orchestrations. In *Third Workshop on Software Evolution through Transformations: Embracing the Change (SETRA'06)*. Electronic Communications of the EASST, 2006.
- [4] E. Bertino, E. Ferrari, and V. Atluri. The specification and enforcement of authorization constraints in workflow management systems. *ACM Transactions on Information and System Security*, 2(1):65–104, 1999.
- [5] W. Binder, I. Constantinescu, and B. Faltings. Decentralized orchestration of compositeweb services. In *Proceedings of the IEEE International Conference on Web Services (ICWS'06)*, pages 869–876, Washington, DC, USA, 2006. IEEE Computer Society.
- [6] Business process execution language v2.0, 2007. <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf>.
- [7] D. W. Chadwick and A. Otenko. The PERMIS X.509 role based privilege management infrastructure. In *Proceedings of the seventh ACM symposium on Access control models and technologies (SACMAT'02)*, pages 135–140, New York, NY, USA, 2002. ACM.
- [8] D. W. Chadwick, W. Xu, S. Otenko, R. Laborde, and B. Nasser. Multi-Session Separation of Duties (MSoD) for RBAC. In *First International Workshop on Security Technologies for Next Generation Collaborative Business Applications (SECOBAP'07)*, Istanbul, Turkey, April 2007.
- [9] G. B. Chafle, S. Chandra, V. Mann, and M. G. Nanda. Decentralized orchestration of composite web services. In *Proceedings of the 13th international World Wide Web*

- conference on Alternate track papers & posters (WWW'04), pages 134–143, New York, NY, USA, 2004. ACM.
- [10] D. K. W. Chiu, S. C. Cheung, S. Till, K. Karlapalem, Q. Li, and E. Kafeza. Workflow view driven cross-organizational interoperability in a web service environment. *Information Technology and Management*, 5(3-4):221–250, 2004.
- [11] D. K. W. Chiu, S. C. Cheung, S. Till, K. Karlapalem, Q. Li, and E. Kafeza. Workflow view driven cross-organizational interoperability in a web service environment. *Inf. Technol. and Management*, 5(3-4):221–250, 2004.
- [12] D. K. W. Chiu, Z. Shan, P. C. K. Hung, and Q. Li. Designing workflow views with flows for large-scale business-to-business information systems. In *Technologies for E-Services (TES'04)*, pages 107–121. Springer, 2004.
- [13] G. Goth. The ins and outs of it outsourcing. *IT Professional*, 1:11–14, 1999.
- [14] P. Grefen, K. Aberer, Y. Hoffner, , and H. Ludwig. CrossFlow: Cross-organizational workflow management in dynamic virtual enterprises. *Int. Journal of Computer Systems Science and Engineering*, 15(5), 2000.
- [15] S. Jablonski, R. Schamburger, C. Hahn, S. Horn, R. Lay, J. Neeb, and M. Schlundt. A comprehensive investigation of distribution in the context of workflow management. In *Proceedings of the Eighth International Conference on Parallel and Distributed Systems (ICPADS'01)*, page 187, Washington, DC, USA, 2001. IEEE Computer Society.
- [16] E. Kafeza, D. K. W. Chiu, and I. Kafeza. View-based contracts in an e-service cross-organizational workflow environment. In *Proceedings of the Second International Workshop on Technologies for E-Services (TES'01)*, pages 74–88, London, UK, 2001. Springer-Verlag.
- [17] M. H. Kang, J. S. Park, and J. N. Froscher. Access control mechanisms for inter-organizational workflow. pages 66–74, 2001.
- [18] Y. Karabulut, F. Kerschbaum, F. Massacci, P. Robinson, and A. Yautsiukhin. Security and trust in it business outsourcing: a manifesto. In S. Etalle and P. Samarati, editors, *Proceedings of STM'06*, 2006.
- [19] G. Karjoth, B. Pfitzmann, M. Schunter, and M. Waidner. Service-oriented Assurance - Comprehensive Security by Explicit Assurances. 2005.
- [20] H. Li and Z. Lu. Decentralized workflow modeling and execution in service-oriented computing environment. In *Proceedings of the IEEE International Workshop on Service-Oriented System Engineering(SOSE'05)*, pages 29–36, Washington, DC, USA, 2005. IEEE Computer Society.
- [21] C. Mohan, G. Alonso, R. Gunthor, and M. Kamath. Exotica: A research perspective ob workflow management systems. *Data Engineering Bulletin*, 18(1):19–26, 1995.
- [22] P. Muth, D. Wodtke, J. Weissenfels, A. K. Dittrich, and G. Weikum. From centralized workflow specification to distributed workflow execution. *Journal of Intelligent Information Systems*, 10(2):159–184, 1998.
- [23] M. Reichert and P. Dadam. ADEPT flex -supporting dynamic changes of workflows without losing control. *Journal of Intelligent Information Systems*, 10(2):93–129, 1998.
- [24] G. Riempp. *Wide Area Workflow Management: Creating Partnerships for the 21st Century*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1999.
- [25] J. H. Saltzer and M. D. Schroeder. The protection of information in computer systems. *Proceedings of IEEE*, 9(63):1278–1308, 1975.
- [26] A. Schaad, V. Lotz, and K. Sohr. A model-checking approach to analysing organisational controls in a loan origination process. In *SACMAT '06: Proceedings of the eleventh ACM symposium on Access control models and technologies*, pages 139–149, New York, NY, USA, 2006. ACM.
- [27] K. A. Schulz and M. E. Orłowska. Facilitating cross-organisational workflows with a workflow view approach. *Data Knowledge Engineering*, 51(1):109–147, 2004.
- [28] Z. Shan, D. K. W. Chiu, and Q. Li. Systematic interaction management in a workflow view based business-to-business process engine. In *Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 7*, page 162.2, Washington, DC, USA, 2005. IEEE Computer Society.
- [29] W. Tan and Y. Fan. Decentralized workflow execution for virtual enterprises in grid environment. In *Proceedings of the Fifth International Conference on Grid and Cooperative Computing Workshops (GCCW'06)*, pages 308–314. IEEE Computer Society, 2006.
- [30] U. Yildiz and C. Godart. Information flow control with decentralized service compositions. In *IEEE International Conference on Web Services (ICWS'07)*, pages 9–17, Washington, DC, USA, 2007. IEEE Computer Society.

APPENDIX

A. BPEL DERIVATION FROM BPG

Algorithm BPEL TREE

```

Input: task  $\tau$ 
Output: BPT  $G'$ 
1: if  $\tau$  is start then
2:   Initialize BPT  $T$  with start;
3: end if
4: if  $P(\tau) \neq \emptyset$  then
5:   for all tasks  $\tau'$  from  $P(\tau)$  do
6:     add to start path  $\chi(\tau')/\tau'$ ;
7:     call BPEL TREE with parameter  $\tau'$ ;
8:   end for
9: else
10:  return;
11: end if

```
