A MVC Framework for Policy-Based Adaptation of Workflow Processes: A Case Study On Confidentiality

Kristof Geebelen, Eryk Kulikowski, Eddy Truyen and Wouter Joosen
IBBT-DistriNet, Dept. Computer Science, K.U.Leuven
Celestijnenlaan 200A
3001 Leuven, Belgium
{*firstname.lastname*}*@cs.kuleuven.be*

***Abstract*—Most work on adaptive workflows offers insufficient flexibility to enforce complex policies regarding dynamic, evolvable and robust workflows. In addition, many proposed approaches require customized workflow engines. This paper presents a portable framework for realistic enforcement of dynamic adaptation policies in business processes. The framework is based on the Model-View-Controller (MVC) pattern, commonly used for adding dynamism to web pages. To enhance reusability, our approach supports separation of adaptation logic from the functional workflow and modularization of workflow tasks in reusable aspects. The main idea is to design a workflow process as a template, where tasks can be specified on an abstract level. Concrete implementations of the tasks, modeled as aspects, are then selected from a library according to a policy-based adaptation logic. This logic is implemented using a general purpose language that offers an extensible and flexible solution to enforce any type of policy. We evaluate by means of a case study on workflow confidentiality to what extent an approach using standards-based technologies allows application-specific adaptation of running workflow instances.**

***Keywords*-service composition; policy-based adaptation; model-view-controller; confidentiality; WS-BPEL**

I. Introduction

Workflow languages focus on combining web services into aggregate services. At this moment, the Web Services Business Process Execution Language (WS-BPEL) [1] has profiled itself as the de-facto industry standard for orchestrating web services. However, this standard exhibits major limitations regarding reusability and flexibility as discussed in several works [2], [3], [4], [5], [6].

A workflow describes a sequence of tasks. Each task represents a coherent set of activities that fulfill a specific functionality. Tasks can be delegated to services and may require human interaction. Most workflow languages assume that the tasks are executed in a static context. However, workflow environments are often dynamic. For example, services can become unavailable, unexpected faults may occur or participating partners in the workflow may not be known upfront, before some tasks are actually executed. Workflow confidentiality is another often recurring requirement that illustrates the need for dynamic adaptation in composite web services. Confidentiality requirements include the enforcement of access control related constraints. For example, a Separation of Duty (SoD) constraint demands that two specific tasks from a workflow are performed by two different users. The identity of the users is usually only known during the execution of the workflow instance. Therefore, dynamic adaptation of the workflow is required to make sure that the second task is performed by a different user than the first. Adaptation of the workflow to the dynamic environment is a different concern than its functional description and thus ideally should be specified separately. Policies can specify how the workflow must behave in a dynamic context. An adaptation policy could state that when a service responsible for a particular task is down, another service providing similar functionality must be used instead. Enforcement of such policies in a functional workflow description results in scattered and tangled descriptions, or may be even impossible to realize.

We identify four important requirements for a realistic solution to dynamic workflow adaptation. First, it is desired that the adaptation policies can be *modeled separately* from the functional requirements to avoid overly complicated workflows. Separation of concerns also promotes the reusability of existing solutions. Second, we want an extensible solution, offering sufficient *flexibility* to implement complex policies that transcend the possibilities offered by traditional workflow languages. Third, our solution must be *portable* between existing workflow engine implementations without invasive modifications. Fourth, to foster *robust* workflows it must be possible to properly handle policies similar to the SoD policy described earlier. Ideally, application-specific adaptation is needed at run-time to orchestrate tasks to users that satisfy the policy. This information is usually only known during the execution of the workflow instance, which means that support for adaptation of running web service compositions is required to avoid failing workflows.

In this paper, we propose a Model-View-Controller framework for policy-based adaptation of workflow processes and show that it supports the four requirements discussed above. The framework treats workflow specifications in a similar way to dynamic web pages, where page content is generated according to the user provided information. Workflows are then tailored from customizable aspects, which represent a coherent set of WS-BPEL activities fulfilling a specific task. The framework supports both deploy- and run-time adaptation of a workflow process. Moreover, in this paper we evaluate by means of a case study on confidentiality to what extent an approach using standards-based technologies allows us to

978-0-7695-4128-0/10 $26.00 © 2010 IEEE
DOI 10.1109/ICWS.2010.81

realize policy-based adaptation and discuss the advantages and disadvantages compared to other approaches.

This paper is structured as follows. Section II introduces the terminology and concepts of confidentiality policies and presents a concrete workflow to illustrate some of the confidentiality policies. Section III presents an overview of the proposed MVC framework and shows how it is realized using standards-based technologies. Section IV illustrates how the framework is used to enforce workflow confidentiality; it motivates the need for workflow adaptation at both deploy- and run-time and illustrates how the framework supports both adaptation strategies. Section V discusses the MVC framework and gives an overview of related work. Finally, section VI concludes the paper.

II. A Case Study On Confidentiality

This section presents a case study on workflow confidentiality. In the first subsection we define the main concepts behind confidentiality. Next, we present an e-health related workflow and its confidentiality requirements.

A. Confidentiality Considerations

Confidentiality has been extensively studied in recent years. An important aspect of confidentiality is controlling access to the resources that contain sensitive information. However, as pointed out by Hammer and Schneider [7], the definition of confidentiality also needs a notion of information flow in order to be complete. In other words, preserving confidentiality requires: 1) **ensuring that information is only accessible to subjects entitled to it**, which can be achieved with access control and encryption mechanisms, and 2) **controlling the information flow** according to the confidentiality policies to ensure that subjects cannot have unpermitted access to information due to its flow.

A workflow consists of a sequence of connected tasks. We define a task as a coherent set of activities fulfilling a specific functionality. A task is executed in the workflow itself or is delegated to a user or organization by a service invocation.

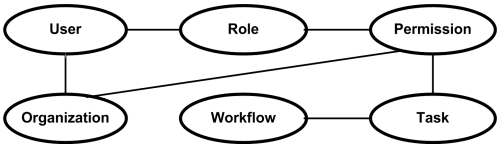

Fig. 1. RBAC Model

For access control purposes we use the RBAC model. It is a frequently used model for controlling access in the context of workflows. More in particular, we assume the model shown in figure 1. This model relates the concepts of organizations, workflows, (delegated) tasks, users, roles and permissions. Users work for an organization. They can have assigned roles and perform tasks when permitted. We extend this model with two types of access control constraints: the **Separation of Duty (SoD)** and **Binding of Duty (BoD)**. For example, we can specify that when task x is performed by a user from organization A, task y also has to be performed by a user from organization A (BoD). Alternatively, we can specify that tasks x and y have to be performed by two different users (SoD).

Tasks are executed within an organization. The organization hosting the workflow can execute tasks itself or delegate them to other organizations or users. Delegating a task usually requires an information flow. An example violation of an information flow policy is when confidential information reaches an organization and is read by a user with a permitted role, however a policy specifies that this information should not be accessible to users affiliated with this organization. In order to enforce information flow policies, our model specifies the assignment of permissions to organizations. A user executing a delegated task needs to have a permitted role and needs to be affiliated with a permitted organization. Tasks that do not involve users only require a permitted organization. Other types of information flow policies usually state constraints on where the information can be stored, over which channels it can be transferred, through which points it can be transferred,

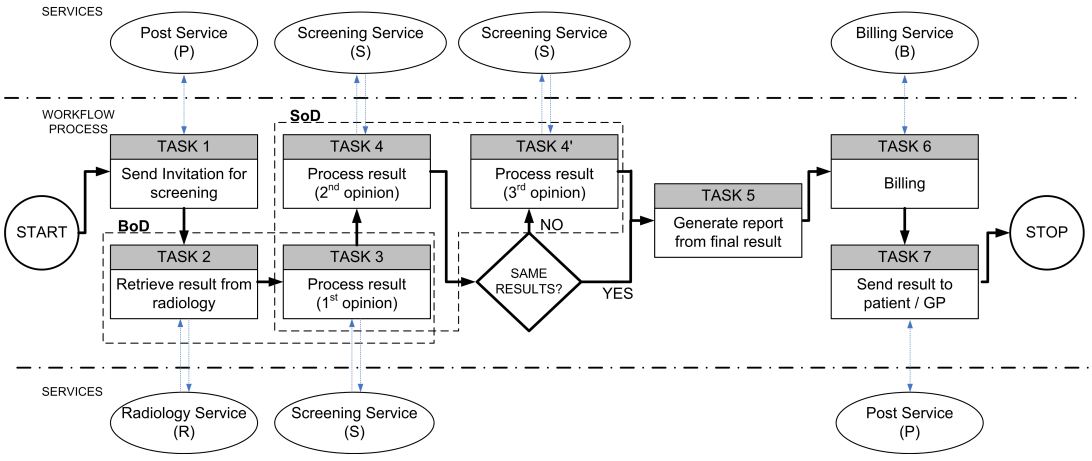

Fig. 2. Example E-Health workflow process

or where it can be viewed. In the context of workflows, the enforcement of these types of policies can usually be reduced to verifying if a particular workflow does not violate these policies, before deploying the workflow.

B. Running Example: An E-Health Workflow

This section presents a case study situated in the health care environment. The case study consists of a workflow that realizes a mammography screening program, initiated by the government, in order to reduce breast cancer mortality by early detection for women above a certain age. The workflow is illustrated in Figure 2.

The first task of the workflow consists of sending out invitations to all women that qualify for the program to let a radiologist take images needed for the screening. Once images are taken, the radiologist uploads them to the system (task 2). A **prevention of acquiring sensitive information constraint** on this result is necessary to avoid a reading of personal information by unentitled persons. Afterwards, the readings can be performed for the particular screening subject. A **BoD** constraint regulates the requirement of assigning participating screening centers. When Task 2 is performed by a radiologist from a specific center (which is not known before the execution, i.e., the screening subject is free to choose the radiology center), the readings have to be performed by the screening center of the corresponding zone (geographical location). This is marked on figure 2 with a dashed rectangle encapsulating tasks 2 and 3. This constraint is also an example of an **information flow policy** on the organizational level, i.e., the involved information may not be accessed in screening centers other than the corresponding one. There are always two independent readings, represented by tasks 3 and 4. In a next step, the two results of the readings are compared. When the results are identical, there is little doubt that the two physicians made the same mistake. Therefore it can be safely assumed that results are correct and the workflow can proceed with task 5. However, when the results are different, a concluding reading is performed (task 4'). A **SoD** constraint implies that in order to be effective, tasks 3, 4 and 4' need to be performed by different physicians within the screening center (dashed line encapsulating tasks 3, 4 and 4'). Once the results of the screening of a particular screening subject are formulated, a report is generated (task 5) and different parties are billed (task 6). Finally, a report is sent to the screening subject and her general practitioner (task 7).

III. A MVC Framework for Policy-Based Adaptation

In this section we present a framework that allows the adaptation of WS-BPEL processes in a dynamic and modular way. The idea is based on similar evolutions in the domain of web design. The intent of web design is to create a website that presents the content to the end users in the form of web pages. To comply with today's expectations of end-users, there is a growing tendency to use dynamic web pages. In contrast to static pages, where the content and layout is not changed with every request, dynamic pages adapt their content on the fly depending on the user's input. We map this concept of dynamic web design to web service composition. The proposed solution is based on the "Model-View-Controller" (MVC) concept.

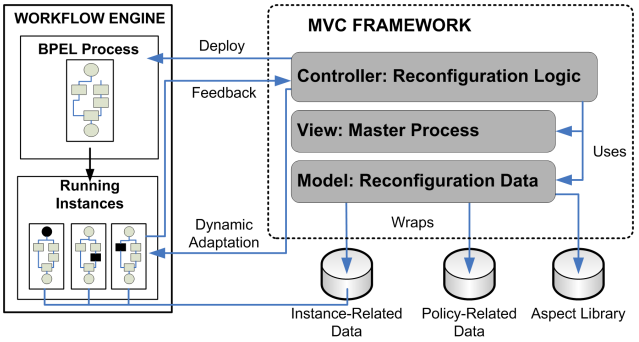

Fig. 3. Framework: Overview

Section III-A gives a high-level overview of the framework and section III-B shows how we implemented a prototype using standards-based technologies.

A. Overview

An overview of our framework is illustrated in figure 3. The framework generates a workflow process according to the adaptation logic specified in the controller component. The resulting process is a standard WS-BPEL process, deployable on existing WS-BPEL engines. Dynamic adaptation of running instances is done according to run-time adaptation logic, for example, based on the execution history of a running instance. During execution, the process instance feeds back to the framework to report its progress. The controller can then decide if adaptation is required. In the context of dynamic web pages, this approach is similar to generating web content based on the user's input. In this analogy, the XML representation of the workflow process can be interpreted as web content.

We discuss the main building blocks of the framework: the master process, the controller and the data model. More details and concrete examples are shown in section IV, where the framework is used to enforce workflow confidentiality.

*1) **Model**:* The model includes the aspect library, the policy- and instance-related data. The library contains aspects for the different WS-BPEL activities that can be modularized as a specific task. All these concrete implementations of a task are bundled in the library and can be reused across workflows. An aspect definition represents a coherent set of basic and structured WS-BPEL activities that realize a particular functionality. Policy-related data specifies properties or parameter values that can be used to evaluate adaptation policies. An example property is the workload of services participating in a particular workflow. Dynamic adaptation of running instances can also depend on instance-related data. This data includes the current value of variables used in a specific workflow and its current executing activity.

*2) **View (Master Process)**:* With the aspect library in mind, a master process can be created. Instead of including all specific implementation details, it is designed as a template. The process is designed like a regular process and specifies the sequence of tasks that need to be executed. When the concrete implementation of a task depends on certain constraints, then only a general reference to the type of the task is included. Binding a concrete aspect to the task reference is done later by the controller according to the adaptation logic.

*3) **Controller**:* The controller contains the logic to decide which aspects are substituted in the master process, i.e. it implements the adaptation policies that depend on the information available through the model. The adaptation policies define a set of context-free and context-sensitive constraints on the specific task implementations that constitute the workflow process. Context-free constraints, where the policy evaluation is done using attributes that are not dependent on the state of the workflow instance, are enforced before deployment using policy-related data. Context sensitive constraints, which require information on the context of the running workflow instance, are enforced at run-time with dynamic adaptation mechanisms using additional instance-related data.

B. Prototype Implementation & Used Technologies

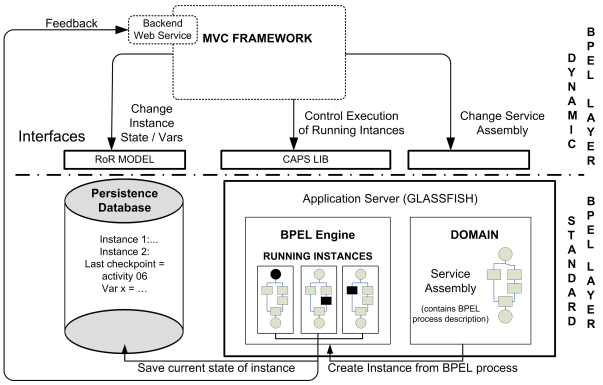

Fig. 4. Prototype implementation

We used standards-based technologies to build a prototype of the framework. The implementation is done on top of the sun-bpel-engine, a component of the Open Enterprise Service Bus. OpenEsb is a Java based open source enterprise service bus. Figure 4 gives a high level overview of how we implemented a dynamic WS-BPEL layer on top of the existing WS-BPEL component framework. The MVC framework we used for the implementation is the "Ruby On Rails (RoR)" framework [8], known for adding dynamism to web pages.

A first step to get a WS-BPEL process up and running in OpenESB is to design the process in a WS-BPEL editor. Once the process is ready, the WS-BPEL module needs to be added to a composite application. The resulting application is packaged as a service assembly and can be deployed and executed on a domain of the application server. The service assembly is copied to the corresponding domain directory on deployment. When a create-instance is triggered by an incoming message, a new running instance will be created on the WS-BPEL engine according to the process description of the WS-BPEL process that was included in the service assembly. The persistence option, which is included in almost every existing WS-BPEL engine implementation, forces the process instance to save its state in the persistence database on crucial points (e.g. after each invoke activity) during its execution.

Our dynamic WS-BPEL layer (top layer in figure 4) interacts with the existing layer through 3 different interfaces. The first allows us to change the WS-BPEL process description within the service assembly. New process instances will be created according to the new description after it is changed. It can be considered as a template from which instances are created. The second interface includes the JAVA CAPS library. Java CAPS stands for Java Composite Application Platform Suite and is composed of several packages that allows to interact with the components of the underlying SOA infrastructure. We use it to interact with the sun-bpel-component. It allows us to restart the engine, recovering workflow instances, etc. The third interface allows us to manipulate the persistence data of running instances. The most straightforward way is to use the model from the RoR framework: tables are mapped on ruby objects which can be manipulated and saved to the database from within Ruby.

Using these insights, we now explain how the dynamic WS-BPEL layer allows us to enforce context-sensitive policies. After a policy sensitive task, the workflow instance feeds back its identifier and the policy rule that must be evaluated using a synchronous invoke on the RoR backend web service. The MVC backend contacts the controller to check for the policy rule for that specific instance. Instance-related data can be retrieved from the persistence database using the RoR model. When a policy rule is not satisfied, the current process can be adapted and rolled back to a previous state. Roll-back is done by changing the last checkpoint in the persistence database (using RoR model) and forcing the instance to restore its state (using the CAPS library). The process is adapted by substituting other aspects in the master process and changing the WS-BPEL description in the service assembly. This allows us to dynamically change the future course of the process instance after a forced recovery when it is required by the policy rule.

IV. Using the Framework to Enforce Workflow Confidentiality

In this section we illustrate how the framework is used to enforce workflow confidentiality in the context of our case study. First, we illustrate example implementations for the building blocks, introduced in section III-A. These building blocks are the main ingredients for the deploy- and run-time adaptation strategies presented next.

```
<aspect name="ScreeningService(Var message)"
    result="Var status">
 <module>
   ...
<invoke partnerLink="SreeningServ"
     portType="ps:screenPT" operation="sent"
     inputVariable="message" outputVariable="status">
 </module>
</aspect>
```

Listing 1. Example aspect definition

```
<process>
  ...
  <%= postService($PsOperationIn.request) %>
   <sequence name="Sequence1">
     <%= radiologyService($RadOperationIn.Request) %>
     <%= screeningService($ScreenOperationIn.request) %>
        ...
      </sequence>
   </scope>
   ...
</process>
```

Listing 2. Fragment of master process

```
module ehealthService
 def postService(order)
   int workloadP1 = Model.getWorkloadP1()
   int workloadP2 = Model.getWorkloadP2()
   if (workloadP1 < workloadP2)
     return aspects.p1(order)
   else
     return aspects.p2(order)
   end
 end
 def screeningService(request)
   if (Model.getZone() == Leuven)
     return aspects.s2(request)
 end
  ...
end
```

Listing 3. Fragment of controller

A. Illustration of the MVC Building Blocks

An organization that hosts a workflow process can have different contacts to delegate a specific task. For example, there are different post services that can be contacted to handle the invitations for a screening. Also different screening centers are able to interpret a mammography of a radiology service. Services can perform the same kind of task, but have different WSDL-interfaces. This means that the workflow implementation will be slightly different for different workflow participants. Aspects can modularize these differences.

Listing 1 shows an aspect definition that allows the invocation of a screening center. The syntax used for defining aspects is based on an existing syntax for describing WS-BPEL modules [3]. A module is a cluster of WS-BPEL activities. Different modules are used when the contained activities that realize the functionality provided by the aspect are scattered throughout the process. Further, listings 2 and 3 show example fragments of the master process and the controller logic respectively. The master process contains regular WS-BPEL activities and parameterizable references to controller methods that implement the adaptation logic. This logic selects aspects that are substituted in the master process.

B. A Deploy- and Run-time Adaptation Strategy

Our motivation for the fact that some confidentiality constraints are ideally enforced at deployment time, while others need to be enforced at run-time is based on assumptions that can be made about the workflow participants. In WS-BPEL, communication with services of organizations or users can be done in a synchronous or asynchronous way. In the case of synchronous communication, a response comes from the same location as the request. If we presume that the location of the endpoint characterizes the identity of the organization or the user, then it is possible to identify the participants of the workflow process before execution. Lower layer security protocols (e.g. SSL) can be used to avoid tampering. However, with asynchronous communication, the response can come from another entity to the one the request was sent to. Matching between request and response can be done using correlation sets. The responder only needs to know the correlation identifier so that the response message can be matched to the corresponding process instance. In this case the workflow needs to authenticate this entity first to determine its role. Policy enforcement can thus only be done at run-time.

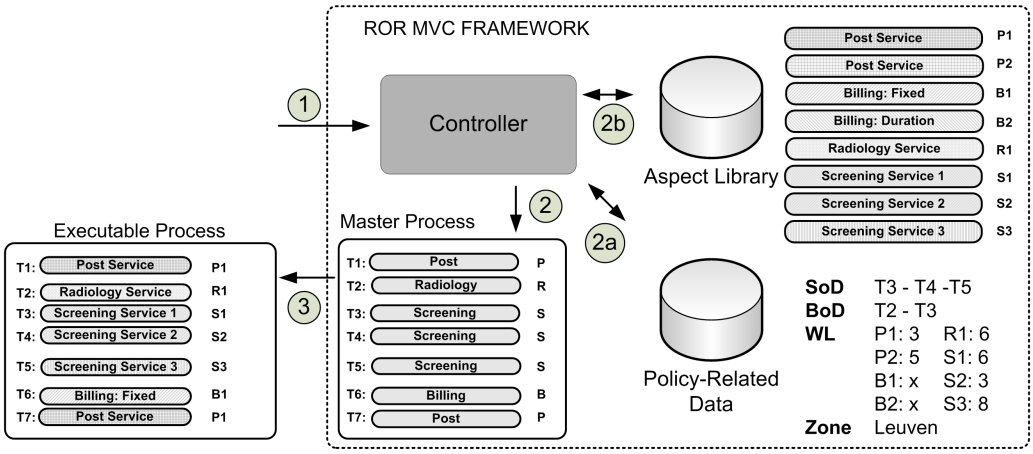

Fig. 5. Deploy-time Adaptation

In the following subsections we present both the simplified case where synchronous communication is used, i.e. the participants of the process can be identified at deployment-time, and the extended case, where run-time identification of the participants is required.

*1) **Deploy-time Enforcement**:* Our goal is to optimize aspect allocation at deployment time, compliant with the context-free policies. Figure 5 shows the different steps of the adaptation process: (1) The controller can be triggered to (re)generate the process. Ideally this is done right before an incoming message triggers the creation of a workflow instance. (2a) The controller interprets partner information and other policy-related data and (2b) selects the corresponding aspect from the library (2) which are then integrated in the master process. (3) The result is a specific executable WS-BPEL process generated by the controller.

We can apply this approach to enforce some aspects of confidentiality at deployment time, presuming the simplified case of synchronous communication where an endpoint of an invoke activity characterizes its identity. A task allocation can be done using adaptation logic that determines the best aspect configuration according to the policies. In listing 3 we showed how **BoD** is enforced by selecting the aspect that invokes the screening center that corresponds to the same zone as the radiology service that is used in the workflow. Analog, the **SoD** requirement can be achieved by selecting aspects invoking endpoints corresponding to different physicians within the same screening center. The **prevention of acquiring sensitive information constraint** and the **information flow policy** is also implicitly enforced because of the deterministic identity assumptions.

*2) **Run-time Enforcement**:* The enforcement of complex access control policies in a more restricted setting, i.e. we do not trust the participating partners, will furthermore require the framework to interpret the access history of objects during execution of an instance of the process.

Figure 6 illustrates how the framework is used to enforce run-time constraints on the e-health workflow. In a first stage, deploy-time adaptation is used to generate an optimal scenario compliant with the context-free constraints (Process Instance 1a). For example, aspect P1 is used to contact the post service to send the invitations. Next, the radiology center, related to aspect R1, sends the result of a screening. Because no assumptions are made about the identity of the endpoint, the use of an authentication mechanism is appropriate. This requires the screening center to authenticate with an identity provider. By appending its proof of identity to the response message, it can be fed back to the MVC framework for validation. Based on this information, together with the history of the process, the MVC framework can regenerate the process instance to be compliant with the context-sensitive constraints.

In our case study, there is a **BoD** between task 2 and task 3 and a **SoD** between task 2, 3 and 4. The initial generated process uses an aspect to contact screening center 2. This is compliant with the constraint policy. So after successful validation of the identity of the response endpoint by the controller, the MVC framework concludes that no regeneration of the process instance is necessary. When the response comes from an unauthorized identity, the instance can be reinitialized on the previous task. The feedback is repeated for the next task, which specifies that a second opinion on the radiology result is required. To enforce the context-sensitive **SoD**, a new process instance is generated (Process Instance 1b), using an aspect to contact screening center 3. The new process instance is initialized on this task and contains the process variables that were set during previous activities. Again, after successful validation of the identity, it seems that both opinions are consistent and therefore task 5 is not required. A new process instance is generated (Process Instance 1c). This instance is not bound to any constraints and can end the workflow.

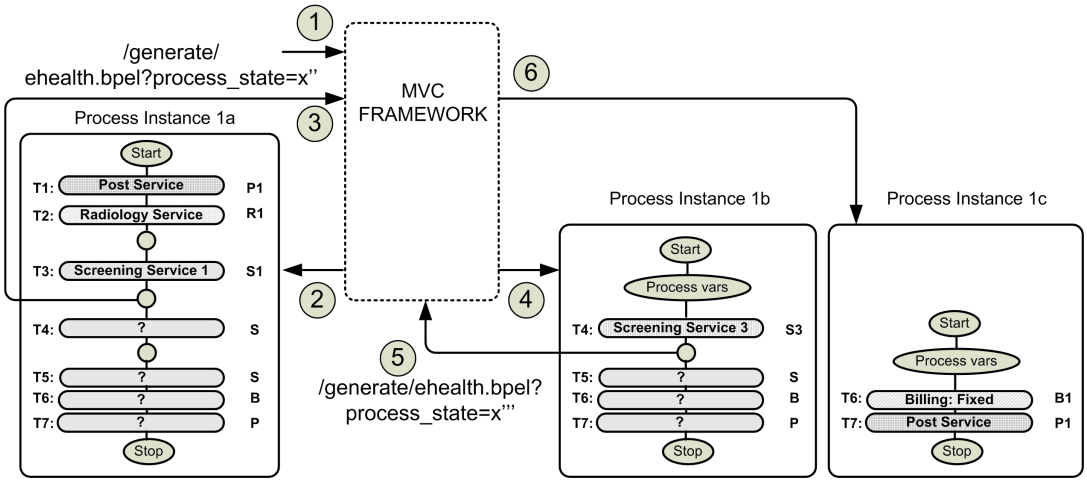

Fig. 6. Run-time Adaptation

Since no assumptions are made concerning the identity of the endpoint, the **prevention of acquiring sensitive information** and the **information flow** policy requires that sensitive information cannot be read by or sent to an unpermitted subject. This restriction can be solved by changing the aspect responsible for contacting a screening center: with a first invocation, the permitted screening center is contacted without adding the mammography result. The screening center responds with a certificate containing its identification and its public key. In a second iteration, the center is sent the mammography result, encrypted with his public key. This implies that only the permitted screening center is able to process the sensitive data by decrypting it using its private key.

V. Related Work & Discussion

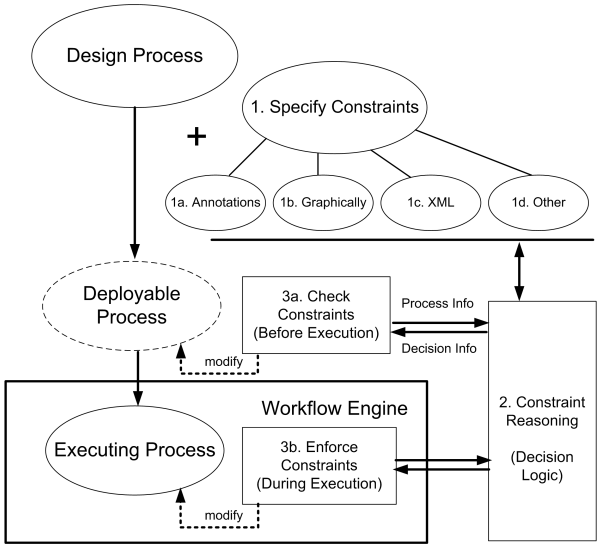

Fig. 7. Recommendation Service: Architectural Overview

Figure 7 illustrates possible stages involved in designing and executing a constrained workflow. Based on this figure we discuss our approach and elaborate on related work that is done in these different stages. A complete and user-friendly solution requires a combination of work done in the different stages.

Business Processes are designed according to the functional requirements. Security constraints are usually modeled separately (Fig. 7: (1)). In the area of security constraint modeling, a lot of work is done on the expression of application security in general. However, specific research in the domain of specifying task-based authorization constraints for workflow models is rather limited. Bertino [10] addresses the expression and evaluation of access control constraints with two formal languages (1c). The first language, RBAC-WS-BPEL, is specified using the RBAC profile of the XACML policy language and extends the WS-BPEL language with support for the specification of authorization information, associated with a business process specified in WS-BPEL. The second is the BPCL language (Business Process Constraint Language) that allows for the description of authorization constraints. In "Modeling of Task-Based Authorization Constraints in BPMN [13]" a graphical approach (1b) is presented. They propose a method to define authorization constraints with a graphical notation language within the Business Process Modeling Notation. The work in the area of constraint modeling is complementary to our work. In our approach, a master process is annotated (1a) with references to the controller. The annotations connect the adaptation logic to specific enforcement points in the workflow, but do not focus on the modeling of constraints. Mapping policies, specified using policy languages to the framework is an interesting track for future work.

In the area of constraint reasoning (2), Hewett [14] proposes practical computational techniques for analyzing SoD with workflows. They present algorithms for generating mutually exclusive roles to enforce SoD and to check if a given RBAC state satisfies a given type of SoD constraint. In our approach, the decision (adaptation) logic is implemented in the controller using a general purpose language. This makes it possible to enforce complex policies that transcend the possibilities offered by the workflow languages, for example, to validate an assertion to reveal an identity. This also allows easy extension of the framework to support any type of the non-functional requirement. For example, load balancing can be done by dynamically changing participating partners in the process. Advanced exception handling and self-healing is possible by changing the process descriptor and recovering process instances according to the new description when faults have occurred.

The main focus of this paper is situated in deploy-time (3a) and run-time (3b) policy enforcement. Related work on the first is done by Hwang [15], who proposes a web service selection approach that chooses a performer for each task in the workflow to satisfy all access control constraints and to increase the chance of completing the entire process in the future. As illustrated in section IV-B1 our approach supports deploy-time policy enforcement. The main advantage is that it introduces no run-time overhead.

We found little related work on practical solutions for the enforcement of access control during the execution of a workflow process (3b). Bertino [10] proposes an XACML based architecture, which manages the execution of business processes subject to the authorization policy and constraints. Access control is realized by intercepting incoming messages of the workflow process. However, it is not clear to what extent the policy decision point has access to the history of the executing process to allow enforcement of context-sensitive constraints. Runtime modification of the executing process is also not possible. A similar approach is done by Wang [17].

A limitation of most existing approaches of adaptive web service compositions, as discussed by Charfi et al. [19], is the tight coupling of the adaptation logic with the execution logic inside the engine implementation. Therefore, the adaptation logic cannot be extended by a third-party. Also, most current approaches do not support dynamic adaptation of running instances or application-specific adaptation features.

Charfi [19] used the dynamic aspect-oriented workflow language AO4BPEL to implement a plug-in architecture for self-adaptive web service composition. Similar to our approach, it supports application specific adaptation scenarios and allows adaptation of running instances by using monitoring and adaptation aspects. The adaptation logic, expressed in AO4BPEL, is weaved in the workflow itself. We use a separate, more flexible general purpose language to express the adaptation logic.

Instead of using an aspect-aware workflow engine, we use standards-based technologies to make our approach portable to existing workflow engines. Our prototype shows that the dynamic WS-BPEL layer is hardly dependent on the underlying engine implementation. Deploy-time adaptation has no dependency on the WS-BPEL engine. With run-time adaptation, the dynamic WS-BPEL layer has a minimal interaction with the underlying layer through the three interfaces discussed in section III-B.

Adding extra dynamism also has a cost. All different stages can introduce unexpected behavior. When aspects are substituted in the master process, WS-BPEL processes can easily become inconsistent. Changing persistent data must also be done carefully to allow successful recovery. A process description cannot be changed before the running state of the instance, otherwise process recovery is not possible since stored variables cannot be matched with the new process descriptor. A feasible approach requires control mechanisms to check for inconsistent behavior.

Another drawback is performance overhead. Introducing feedbacks in the process, enabling workflow persistence and adapting instances slows down the execution time. This overhead is negligible for long-running process instances that can exist for several days, weeks or even months. For real-time processes we advice to use the static approach where policy checking is only done at deployment time and thus no run-time delays are introduced. On the other hand, dynamic adaptation can also improve performance. There is a gain as workflows are not interrupted or need to start all over because of unexpected behavior.

VI. Conclusion

In this paper we have presented a Model-View-Controller framework that allows policy-based adaptation of WS-BPEL processes. Based on the properties of the environment and the running workflow instances (Model) this framework can dynamically adapt a workflow instance. A workflow is designed as a master process which represents a template where tasks can be specified on an abstract level (View). Concrete implementations, modelled as aspects, are then selected by the adaptation logic according to the policy (Controller). Our framework, realized using standards-based technologies, supports *modularization* of tasks in reusable aspects, has *flexibility* to support complex policies as the enforcement logic is implemented in a general purpose language, is *portable* to different execution environments as it is independent from the workflow language, and it allows the design of *robust* workflows as they can be adapted and rolled-back.

References

[1] Web Services Business Process Execution Language Version 2.0, April 2007, OASIS Technical Committee, http://docs.oasis-open.org/wsbpel/.

[2] Charfi, A., Mezini, M., "Aspect-oriented web service composition with AO4BPEL," , Proceedings of the 2nd European Conference on Web Services (ECOWS 2004), Erfurt, Germany, Springer-Verlag, 2004, pp. 168-182.

[3] Braem, M., Verlaenen, K., et al., "Isolating process-level concerns using Padus," Proc. of the 4th International Conference on Business Process Management (BPM2006), Vienna, Austria, Springer , 2006.

[4] Kloppmann, M., Rickayzen, A., et al., "WS-BPEL Extension for Subprocesses - BPEL-SPE.", A Joint White Paper by IBM and SAP, 2005.

[5] Akram, A., Meredith, D. and Allan R.: Evaluation of BPEL to Scientific Workflows. Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06).

[6] Braem, M., Joncheere, N., Geebelen, K. And Verlaenen, K., "Guiding aspect-oriented service composition in WS-BPEL and Padus (Demonstration), " Proceedings of the 6th international conference on aspect-oriented software development, March 2007.

[7] J.H. Hammer and G. Schneider. On the Definition and Policies of Confidentiality. Third International Symposium on Information Assurance and Security (IAS), pages 337-342, 2007.

[8] Thomas, D.,Hansson, D.,Breedt, L. and Clark, M.,"Agile Web Development with Rails," 2nd Edition (2007).

[9] 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, vol. 2, no. 1, 1999, pp. 65 - 104.

[10] E. Bertino, J. Crampton, and F. Paci, "Access Control and Authorization Constraints for WSBPEL," Proc. IEEE International Conference on Web Services (ICWS 2006), 2006, pp. 275 - 284.

[11] Botha R. A. and Eloff J. H. P., "Separation of duties for access control enforcement in workflow environments", IBM System Journal, 40 (3), 2001, pp. 666-682.

[12] M. Srivatsa, A. Iyengar, T. Mikalsen, I. Rouvellou, and J. Yin, "An Access Control System for Web Service Compositions," Proc. IEEE International Conference on Web Services (ICWS 2007), 2007, pp. 1-8.

[13] Wolter, C., Schaad, A., "Modeling of Task-Based Authorization Constraints in BPMN," In: Alonso, G., adam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, Springer, Heidelberg, 2007, pp. 64-79.

[14] Hewett, R., Kijsanayothin, P., and Thipse, A., "Security Analysis of Role-based Separation of Duty with Workflows," In Proceedings of the 2008 Third international Conference on Availability, Reliability and Security (March 04 - 07, 2008). ARES. IEEE Computer Society, 2008, Washington, DC, 765-770.

[15] San-Yih Hwang, Chuan Yin, Chien-Hsiang Lee, "Selecting Web Services and Participants for Enforcing Workflow Access Control," hicss, 42nd Hawaii International Conference on System Sciences, 2009, pp.1-10.

[16] J. Mendling, K. Ploesser, M. Strembeck, "Specifying Separation of Duty Constraints in BPEL4People Processes," In: Proc. of the 11th International Conference on Business Information Systems (BIS), Lecture Notes in Business Information Processing (LNBIP), Vol. 7, Springer Verlag, Innsbruck, Austria, May 2008.

[17] Wang, X., Zhang, Y., Shi, H., Yang, J.: BPEL4RBAC: An Authorisation Specification for WS-BPEL. In: Bailey, J., Maier, D., Schewe, K.-D., Thalheim, B., Wang, X.S. (eds.) WISE 2008. LNCS, vol. 5175, pp. 381-395. Springer, Heidelberg (2008)

[18] Geebelen, K., Michiels, S., and Joosen, W. 2008. Dynamic reconfiguration using template based web service composition. In Proceedings of the 3rd Workshop on Middleware For Service Oriented Computing, Leuven, Belgium, 2008, MW4SOC '08, 49-54.

[19] Charfi, A., Dinkelaker, T., Mezini, M., "A Plug-in Architecture for Self-Adaptive Web Service CompositionsWeb Services", IEEE International Conference on Web Services (ICWS 2009), 2009, pp. 35 - 42 .

[20] Courbis, C., Finkelstein, A., "Towards aspect weaving applications," Proceedings of the 27th international conference on Software engineering, New York, ACM Press, 2005, pp. 69-77.