**TAS[3] Deliverable D3.1 - final:**
Design of a semantic underpinned, secure & adaptable process management platform (final)

Accompanying Letter to the Reviewers

In the report of the TAS[3] review on March 11, 2011, the project reviewers had comments on Deliverable D3.1 (3) "Design of a semantic underpinned, secure & adaptable process management platform (3)" delivered in M24. The TAS[3] Consortium has addressed the reviewer comments as follows.

*a) The efficiency and the limitations of the proposed approach are not addressed.*

The overall approach is presented in the Sections 4.2 and 4.7, as well as in the publications "The Architecture of a Secure Business-Process-Management System in Service-Oriented Environments." and "Secure Business Processes in Service-Oriented Architectures – a Requirements Analysis". The first paper has been presented at the ECOWS2011 Conference, a selective venue with 24% acceptance rate only, the second paper at the ECOWS2010 Conference with 19% acceptance rate. In Section 4.7.8 there now is an explicit discussion of the efficiency and the limitations.
Another deliverable where these issues are addressed is D3.3, Section 6, from the perspective of deploying our components to three pilot applications and to the TAS3 security framework.

*b) The management of different security policies (annotations, data privacy, core security infrastructure) remains unclear.*

The relationship between the various concepts is described in Section 4.5 and in a slightly abridged version in the paper "Modelling and Transforming Security Constraints in Privacy-Aware Business Processes" which has been accepted for the SOCA Conference 2011 (http://www.ipd.kit.edu/~muelle/SecLangTransfBPMS.pdf) and is as follows:

- The process modeler specifies all important security characteristics of the process by means of BPMN annotations.
- Annotations ultimately must be mapped to something executable: Depending on the annotation, there are three different mapping targets altogether. (Note that we are first to have observed this distinction, it has not been made earlier in the literature.) The different targets are as follows: (1) process fragments inserted into the process, (2) parameter values (which we have dubbed configurations) of existing security components, (3) policies (which eventually are processed by standard components such as vanilla PDPs).

Orthogonally to the scheme just sketched, there are external policies, e.g., access policies for data specified by the data owners, privacy policies of users, or sticky policies. Our PEP forwards this kind of policy to the respective components, to, say, PDPs (i.e., PDPs not developed within WP3) or components that are part of the extended architecture designed by us such as the KENT sticky policy handler.

*c) The adaptation of business processes remains limited.*

The description of work does not announce an all-encompassing approach regarding process adaptation. According to it, the plan has been to address process adaptation in the context of security. We strive for adaptation mechanisms for business processes that are useful from a security perspective. Given the experiences we have gathered with the pilot applications in TAS3, it has turned out to be sufficient to adapt processes by inserting process fragments. This is described in Section 4.2.3, and we make use of this adaptation feature when it comes to the transformation of security annotations to something that is executable. The adaptation approach is described in Section 4.6, and we have revised it to clarify how adaptation mechanisms should look like to cope with the requirements of secure business processes.

As a side issue, the previous review report has stated that KIT has not demonstrated 'secure business process adaptation' at the last review meeting. This is not correct. We have demonstrated security-enhanced business-process management, which is based on process adaptation. The two pilot demonstrators presented have worked with security-annotated business processes. They have included invocations of security components of the TAS3 framework and process-specific security functionality within the process.

The WP03 Team

**SEVENTH FRAMEWORK PROGRAMME**
**Challenge 1**
**Information and Communication Technologies**

**Trusted Architecture for Securely Shared Services**

| | |
|---|---|
| **Document Type:** | Deliverable |
| **Title:** | **Design of a semantically underpinned, secure & adaptable process-management platform** |
| **Work Package:** | WP3 |
| **Deliverable Nr:** | D3.1, third iteration |
| **Editor:** | Jutta Mülle, KIT – Karlsruhe Institute of Technology |
| **Dissemination:** | PU |
| **Preparation Date:** | November 03, 2011 |
| **Version:** | 3.4 |



SEVENTH FRAMEWORK PROGRAMME

# The TAS$^3$ Consortium

| | Beneficiary Name | Country | Short | Role |
|---|---|---|---|---|
| 1 | K.U.Leuven | BE | KUL | Coordinator |
| 2 | Synergetics nv/sa | BE | SYN | Partner |
| 3 | University of Kent | UK | KENT | Partner |
| 4 | University of Karlsruhe | DE | KARL | Partner |
| 5 | Technische Universiteit Eindhoven | NL | TUE | Partner |
| 6 | CNR/ISTI | IT | CNR | Partner |
| 7 | University of Koblenz-Landau | DE | UNIKOLD | Partner |
| 8 | Vrije Universiteit Brussel | BE | VUB | Partner |
| 9 | University of Zaragoza | ES | UNIZAR | Partner |
| 10 | University of Nottingham | UK | NOT | Partner |
| 11 | SAP Research | DE | SAP | Project Mgr |
| 12 | Eifel asbl | FR | EIF | Partner |
| 13 | Intalio Ltd | UK | INT | Partner |
| 14 | Risaris Ltd | IR | RIS | Partner |
| 15 | Kenteq | BE | KETQ | Partner |
| 16 | Oracle | UK | ORACLE | Partner |
| 17 | Custodix nv/sa | BE | CUS | Partner |
| 18 | Medisoft bv | NL | MEDI | Partner |
| 19 | KIT | DE | KARL | Partner |
| 20 | Symlabs SA | PT | SYM | Partner |

# Contributors

| | Name | Organisation |
|---|---|---|
| 1 | Jutta Mülle (1st, 2nd, and 3rd iteration) | KARL |
| 2 | Jens Müller (1st, 2nd, and 3rd iteration) | KARL |
| 3 | Thorsten Haberecht (2nd and 3rd iteration) | KARL |
| 4 | Silvia von Stackelberg (3rd iteration | KARL |
| 5 | Ioana (3rd iteration) | VUB |
| 6 | Quentin Reul (1st iteration) | VUB |
| 7 | Jeroen Hoppenbrouwers (1st iteration) | SYN |
| 8 | Arnaud Blandin (1st iteration) | Intalio |
| 9 | Alex Boisvert (1st iteration) | Intalio |

# Contents

# List of Figures

# List of Tables

# Executive Summary

In TAS$^3$, any communication is subject to specified policies. Compliance is checked for every request and every reply, both at the service requester and at the service provider side.

Business process management provides a flexible approach for defining and running applications in service oriented architectures with web services as basic building blocks. A business process orchestrates web service calls, human interactions via web service interfaces and reactions of external events providing an explicit specification of the flow. The security aspect in business processes relates to policy enforcement points which will intercept any web service call to or from the business process and enforce any applicable policy. These policies are specific to business processes in that way that they can refer to properties of the process model or the process instance in question. Such properties may be the execution status of the process instance (such as activities waiting for execution, values of internal variables or the execution history), the security context of the process instance, the roles and resources assigned to the process, or the description of the process model, e.g., its privacy policy.

The topic of work package three is the support of adaptive, secure business processes in the TAS$^3$ framework. This document describes the conceptual design and basic components of the system architecture for business-processes support.

The TAS$^3$ architecture is based on executing business processes including web-service invocations. Therefore, we first describe the concepts for business process modelling and execution in a service-oriented environment and an open source software system with state of the art technology. Using an example process of the employability domain, we analyze requirements in detail and discuss them for the core topics of WP3, i.e. secure processes, secure adaptation of processes, semantics of business processes, and the software architecture. Following the requirements analysis, conceptual design yields mechanisms and concepts for secure business processes. Further, it features concepts for security-guided adaptations of the schemas and content of running process instances, e.g., selecting in a specific (security-related and process-specific) context different services with respect to their security properties. Modelling business processes allows to handle security specifications at the business level as well. We provide mechanisms to transform its information to security specifications on a policy and executable level. In order to support the modelling of security specifications using adaptability of processes, semantics specifying the security context of processes will underpin the business process management.

We achieved results in detailing the business-process specific security mechanisms, in particular we have introduced a business process security policy and its relationship to the other components. With respect to adaptability we have a concrete comprehensive overview about the techniques of our adaptation approach and added mechanism to adapt processes with of use of process patterns and of adapting processes on the instance level. Further, we applied adaption and used it for supporting the security modelling level for business processes. Our research strongly focuses on the relationship between adaptation and process security concepts. Finally, a lot of results come with our approach for modelling security for business processes which arose from a basic approach and the validation results of the demonstrator. The security annotations of process models now build a comprehensive set of security rules, and our approach for transforming the specifications to the enforcement level. For that adaptation techniques provide an essential part. To provide better usability, we propose an ontology based security annotator supporting the modellers with help for specifying the security annotations.

We have described components implementing the conceptual design in Deliverable D3.2 ([2]) and have applied them to selected pilot applications, see Deliverable D3.3 ([3]). D3.3 also contains a lot of modelling examples together with realizing them in the security framework of the project and in particular with the business-process specific components.

### 0.0.1 Reading Guide

This report contains some introductory material. Below we give some recommendations how to read this report, for different groups of readers.

Individuals having already read a previous version of this document may want to focus on the following

changed and enhanced sections:

In Section 4.2 we introduce our approach to model security together with business processes. We have developed annotations which define security properties of the business processes. This is the final language resulting from refinements according to our experiences with modelling demonstrator applications of the e-employability domain and a BTG application for the e-health domain, and of a user study on modelling security of business processes with our language (see Deliverable D3.3 [3]). To this end, the modelers like business analysts who are familiar with the application level get support to understand and define security constraints of an application which is realized as a business process. Section 4.2.2 gives an introduction into the language of security annotations. We have newly added the description of the language as Appendix 6. How to transform the security annotations to the enforcement level is subject of the refined Section 4.2.4. Section 4.6 has been reorganized and features concepts on adaptability support for business processes, especially, how to use process adaptation for security aspects. The chapter gives an overview about adaptation methodologies and fundamental techniques which we propose to use for secure business process management. To this end, we particularly provide in Subsection 4.6.4 an adaptation method of business processes using subprocesses of a repository of security-related process patterns. Sections 4.4 and 4.5 describe the relationships between security constraints of business process models, configuration of secure process execution and the BP-specific security components. Also, we discuss the interplay with the core security infrastructure, e.g., with other security and trust policies. In Section 4.2 we introduce explicitely our overall approach, see also [1]. In Section 4.8 there now is an explicit discussion of the efficiency and the limitations.

Readers familiar with business-process-management concepts and systems can skip subsection 2.1, as well as subsection 2.2 if readers have some insight into the Intalio framework.

Readers whose interest is focused on the main topic of work package 3, security management for processes, might want to start with chapter 4 containing our main part of the report, namely the conceptual design in detail and the integration into the TAS$^3$ security and trust architecture, and then continue with chapter 5 describing the mapping to some of the concepts of chapter 4 to systems and components for implementation.

For topics of particular interest, the reader is encouraged to go back to chapter 3 that meticulously motivates and gives way to the conceptual design. Section 5.1 presents the integration of the Business Process specific security components into the TAS$^3$ architecture described in [4]. Section 4.7 describes the results of the conceptual design of security mechanisms specific to business processes in more detail. In particular, Section 4.4 presents the conception of a business process security policy, which contains security constraints specific to business processes, namely taking into account process elements like tasks, roles. In Section 4.2 we introduce our approach to model security together with business processes. A process meta model allows to use annotations which define security properties of the business processes. To this end, the modelers like business analysts who are familiar with the application level get support for understanding and defining security from an application. Section 4.2.2 describes in detail the security annotations and gives examples. The complete description of the security annotation language contains Appendix A. How to tranform them onto the enforcement level is subject of Section 4.2.4. In Section 4.2.5 we introduce an annotator, i.e. a component that supports users (modellers) to forulate security annotations by a knowledge base based on an ontology with business-process-specific security concepts. Section 4.6 features concepts on adaptability support for business processes, especially, how to induce and guide process adaptation by security aspects. The chapter gives a detailed overview about adaptation methodologies which we propose to use for secure business process management. To this end, we particularly provide in Subsection 4.6.4 an adaptation method of business processes using subprocesses of a repository of security-related process patterns. The enforcement of the refined business-process security is subject to the implementation chapter 5. Section 5.4 provides a newly developed technique to handle intervals during process execution and prepare this information for use in the context-specific security components of business process management.

# 1 Introduction

## 1.1 Scope and objectives

TAS$^3$ has the goal to provide a next-generation trust & security architecture that

- meets the requirements of complex and highly versatile business processes,
- enables the dynamic user-centric management of security and trust policies, and
- ensures end-to-end secure transmission of personal information and user-controlled attributes between heterogeneous context-dependent and continuously changing systems.

TAS$^3$ aims at developing a Trust Network (TN) where parties, e.g., Service Providers (SPs) and users, can interact with each other securely. Thus, we must provide the means to explicitly handle interactions of services, processes and data on the TN. In principle, business processes offer this functionality. The topic of work package three is the support of adaptive, secure business processes in the TAS$^3$ architecture. Specific objectives are:

- provide security mechanisms to handle authorisation and access control of workflows and their contexts, e.g., human participants involved and underlying application data
- provide security mechanisms using adaptations of live processes
- support the security requirements of adaptive workflow management in trusted distributed personal information processing and eHealth scenarios
- have all relevant business process descriptions annotated to a common, agreed upon semantic knowledge structure in line with partners.

This report starts by discussing the requirements of TAS$^3$ scenarios regarding business-process-management support. It summarizes and refines the requirements of Deliverables D1.2 [5] and D1.4 [6], and by pointing to issues left open by existing systems and approaches. Then the conceptual framework of business processes for TAS$^3$ is described.

The TAS$^3$ architecture is based on executing business processes using web services, user interactions during process execution and processing underlying data. Usually, there exist several alternative services and data sources that are adequate to support a particular activity in a business process. It should not be necessary to pin down in advance which activity/service will be used during execution. In line with the overall direction of TAS$^3$, we put much emphasis on processes that handle privacy-relevant data and personal information. We will exemplify this by using e-portfolios used for employment and training services and the health status of individuals in e-health applications. The services are provided in a distributed environment and are flexibly offered via web services with many participants involved, at distributed sites.

Thus, we first need to provide a process-modelling and execution framework in a service-oriented architecture. Following the requirements according to our analysis, the conceptual design then provides mechanisms and concepts for secure, privacy-preserving business processes on the modelling and the execution level. The architecture developed defines the components for security enforcement of business processes in line with the fundamental TAS$^3$ security and trust architecture. The focus of the conceptual model lies on security support in business processes. Further, the conceptual design introduces concepts for altering and adapting schemas of process models, e.g., to select appropriate services with respect to their security properties or quality properties according to the requirements of the data processed.

In order to support the modelling of security specifications and the adaptability of processes, ontology methods semantically underpin the business processes with ontology-based descriptions and annotations of the business-process components. Ontologies of business processes contribute by adding security issues

at the modelling and execution level in order to yield processes with a specific security level and allow for process adaptation using security-related process fragments, in particular.

An example process from the employability scenario serves as the basis for further validation of the research results. The process model describes the result of modelling one of the business processes of the employability domain. The goal is to set up a real-world business process to validate the concepts and the framework for business-process support we will have developed. In this report the real-world business process helps to analyse and refine requirements and to get started with the system-architecture design and the conceptualization of adaptive and secure processes.

The conceptual design comprises mechanisms for support the phases of a business-process from modelling via transformation to execution of the process. For modelling we have developed a security annotation language for BPMN process models, which allow BP analysts and designers to specify BP-specific security constraints and relate them directly to the BPMN elements affected. This is designed as a plugin to the graphical modelling tool. Starting with these security annotations, we provide transformations to certain targets of the execution level of an BPMS. To this end, we are using an adaptation approach for business processes, which transforms part of the annotations by security-specific process fragments into an enhanced business process model. Another target consists of rules of a business process policy, which will be checked by the policy decision point of the BPMS. Finally, the last target are the components of the security framework storing and handling context-specific parameters for calling the security components by the policy-enforcement point of the BPMS. The design of the secure BPMS for executing and enforcing the secure business processes with their security constraints comprises an architecture of the secure BPMS and BP-specific security components. These components interact with the project security framework in a well-defined way. For conceptually designing the execution components, we have formalized the functionality and refined the design of the components. The conceptual design concludes with regarding the efficiency of our overall approach and limitations given.

In this report we introduce an overview and a first design of the implementation of security components for business process management. For details and further development we refer to the Deliverable D3.2 [2], the description and documentation of the software implementing the components of a secure business-process-management system. Finally, we deploy the secure business-process-management system to selected pilot applications, see Deliverable D3.3 ([3]). D3.3 also contains several business process models together with their implementation in the security framework of the project and in particular with the business-process specific components.

## 1.2 Document organisation

The rest of the document is organised as follows.

Chapter 2 is a brief introduction to business process management, respective architectures, modelling languages, and to the concrete business process management system in use (Intalio).

Chapter 3 describes an example scenario, i.e. a business process of the employability application area, and introduces the modelling method used for business-process design. Then it presents the results of a detailed requirements analysis for the different research threads in this work package, i.e., security management for business processes, secure adaptation of processes during execution and the influence of security on process adaptation, providing semantics of security in business processes in order to underpin adaptable secure processes with ontologies, and requirements on the security architecture.

Chapter 4 presents the conceptual design of security management for business processes up to now. It describes the architecture of the security support for business processes with the components and the enforcement process, and the alignment with the TAS$^3$ security architecture. Further, it contains an approach of using the business process level to define security rules which allow configuring security components in an automatic way, e.g., policies or input for policy-enforcement points. As a next step, it provides an overview of the use of ontologies for secure business-process management, and the secure process-adaptation approach. Finally, our concepts for secure business processes are presented in detail.

In Chapter 5 the realisation of the concepts presented in Chapter 4 are investigated, and it is described how they map to components and by large how they will be implemented. The chapter starts by describ-

ing the integration with the TAS$^3$ trust and security architecture and with the applications. The main part contains our efforts to realize security concepts for processes. Further on, it describes an approach to business-process related security modelling together with the modelling of the business process for security configuration of the business execution and security and enforcement layer, and fundamental mechanisms to handle process adaptations to support security, i.e. security-aware process adaptations. The implementation is elaborated in Deliverable D3.2.

Chapter 6 gives a conclusion.

Appendix A contains the description of our security annotation language for business processes in detail.

# 2 Fundamentals of Business Process Management

This chapter gives an introduction to business process management and the business process management framework of Intalio. Readers familiar with business process management can skip Subsection 2.1 those with some familiarity into the Intalio framework can skip Subsection 2.2.

## 2.1 Business Process Management

One objective of TAS$^3$ is to create a trusted architecture for securely shared services. Thus, a crucial issue is the successful interaction of (web) services of different owners. The process of coordinating the flow of activities and data is known as orchestration. The result is a business process managed by a business process management system. In the report, we use the terms business processes and workflows as synonyms, if not otherwise stated. The term "workflow" usually puts emphasis on flows of activities which also contain human activities and coordinate human interaction with the process as well. But nowadays business process management usually comprises human interaction, as this feature becomes more and more important for businesses.

For the TAS$^3$ project, we follow a standards-based approach. Thus, BPEL [7] is the language of choice to model executable business processes, as is BPMN [8] for modelling them. Figure 2.1 is a rough schema describing the levels of business processes.

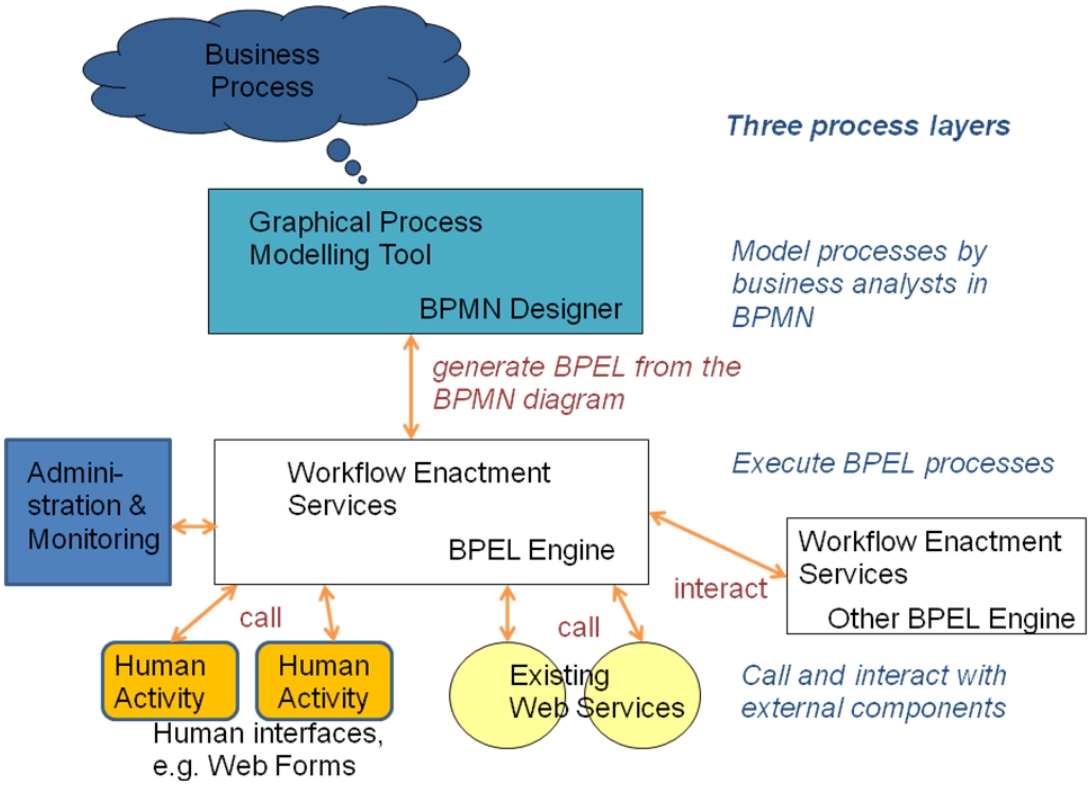


**Figure 2.1: Business Process Layers**

Deploying a new business process or changing an existing one consists of several steps: First, one has to create a (formal) model of the process (or change the existing one). This is the task of business analysts, i.e. on the business level. Then a transformation of the model into an executable business process model takes place. After augmenting this result with additional implementation parts, such as concrete web services descriptions, workflow data, and web forms for human activities, the deployment on a business process management server yields a comprehensive executable process.

Based on certain events, e.g., incoming web-service calls, time events, or interactions with humans, the

server creates new instances of the process and executes them. After instatiating an executable process it is run on workflow enactment services provided by a BPEL engine. Reactions to events are explicitly modelled in the business process model.

The business process orchestrates calls to web services or other business processes and interactions with so-called human tasks. Human tasks provide a user interface for user interactions e.g. using a web form.

The administration and monitoring component allows administrating the execution of business processes, e.g. to stop or cancel a process execution. Usually this component also comprises monitoring facilities.

The Workflow Management Coalition [9] provides a standardized component model of Workflow Management shown in Figure 2.2. It also contains an administration and monitoring component, which we need for business process management in TAS$^3$ as well. Intalio provides a management console that includes the handling of human activities. Human activities correspond to the user client applications and worklist handler. The applications invoked, linked via tool manager, are web services.

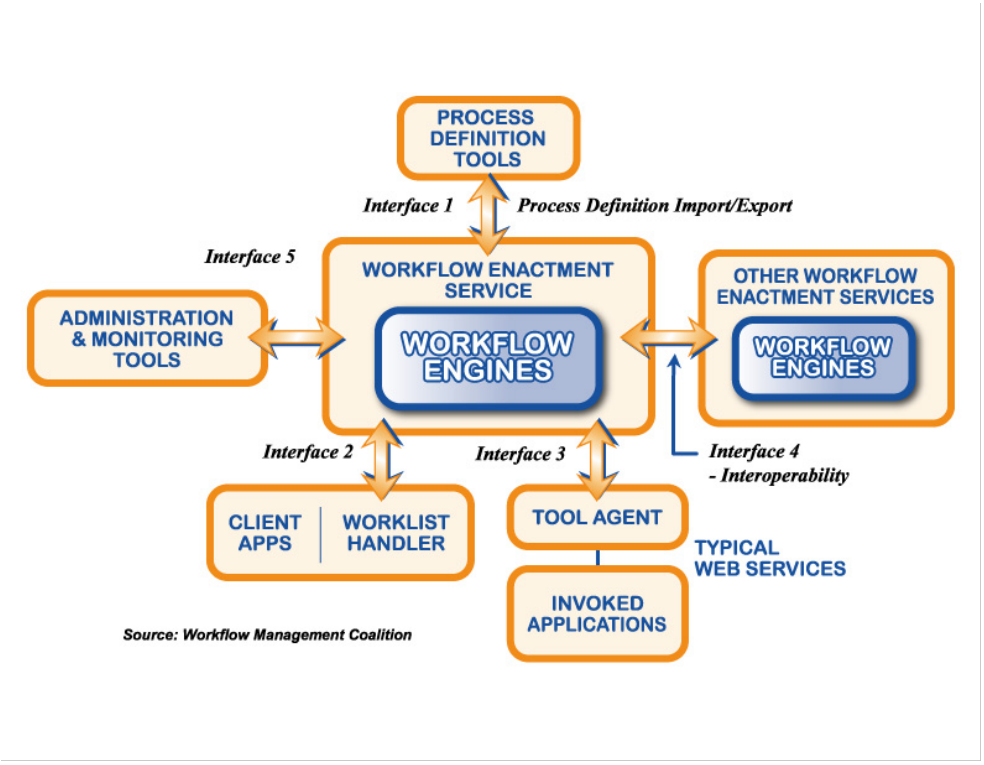


**Figure 2.2: Reference Model of the Workflow Management Coalition**

Thus, the deployment of a process consists of several steps, with distinct components responsible for each step, as already described at the beginning of this section. The following list relates the Intalio components to these steps:

1. The modelling of the process (as a BPMN model enhanced with annotations to support the execution) is the responsibility of a BPMN modelling tool, such as Intalio—Designer.

2. The next task is the translation of the graphical BPMN model into an executable form and the deployment of the resulting BPEL process and, e.g., XForms for specifying human user interfaces onto the BPEL execution and workflow engines, respectively. Intalio—Designer does this as well.

3. Finally, the resulting process description is instantiated and executed. The process calls web services as modelled in the business process model and interacts with humans, e.g., by human activities, and may interact with other business process execution engines. In the Intalio—BPMS system, this task is split between the Apache Ode server, which handles the execution of the actual BPEL process instance, and the Tempo engine, which handles user interaction with web forms. An Intalio process

interacts with other business processes, also with processes running on other workflow engines via web service calls.

Any comprehensive approach on integration of security and adaptability of business processes must address all three phases and consequently all of the corresponding software components.

### 2.1.1 Languages for Modelling Business Processes

The Business Process Modelling Notation (BPMN, see [8]) specification provides a graphical notation to express business processes in a business-process diagram. The objective of BPMN is to support business-process management for business process management experts and users involved in the process. To this end, it provides a notation that is intuitive to business users yet able to represent complex process semantics. The BPMN specification also provides a mapping between the graphical notation and the underlying execution language for business processes, particularly BPEL [7] (Business Process Execution Language).

BPMN consists of the following categories of elements: activities, gateways, connecting objects, annotations, data, and pools (with lanes).

Activities comprise tasks as basic activities and compound activities, which contain a BPMN subprocess and can be, say, an iterative activity or a transactional activity.

Gateways provide routing elements to join or split flows of activities, if applicable under specific conditions.

Connecting objects relate the other elements to each other, e.g., to define sequences of activities, gateways or events. Messages are needed to combine elements of distinct pools, defining a message flow.

Annotations allow adding data objects and arbitrary descriptions to BPMN elements.

Pools are structural units each containing a BPMN (sub)process. E.g., a pool can be used to represent a role or an organization. Interaction between elements of different pools is only possible via messages. This reflects the service calls between distributed nodes, as, say, different organizations or roles.

BPMN is used to specify a conceptual model of the business process by business analysts. What we need next is a language which is executable, i.e., for which a workflow execution machine exists. In the context of service-oriented frameworks this actually is BPEL, the business process execution language (see [7]). It is a language that allows for business-process logic to be expressed decoupled from existing software, yet tied to external software through (web) service invocations. This reduces and potentially eliminates the need to code business-process logic in a traditional programming language, such as Java, C/C++, etc. This clear separation of concerns between business processes and software services makes process logic (e.g., workflow management) simpler, more focused and easier to manage. BPEL provides modelling elements which allow transforming BPMN to a BPEL notation. However, there exist various transformations of one BPMN schema to BPEL schemas. This is because BPEL offers several possibilities to express the same concept of a BPMN model. The BPMN standardization proposes one possible transformation which could be used, but in practice each business process management system takes its proprietary transformation. So a pragmatic way is to rely on the transformation provided by the business process management system used, in our case Intalio [10].

In TAS$^3$ business analysts are using BPMN to model the business processes. Data definitions, usually in XML, and web service descriptions in the XML-based WSDL format [11] enhance the BPMN process model as well as the tool-supported design of the web interfaces for users interacting with the business process. Having all this information then allows to automatically generate executable BPEL code. The BPEL execution engine can execute the business process instances with this code. So BPEL is not visible for the TAS$^3$ users, they do not have to deal with it. To implement process security enhancements and adaptation of processes we will have to work with BPEL as well, e.g., to transform security enhancements of the BPMN model to an executable BPEL representation or to handle process adaptations.

### 2.1.2 BPEL4People - Enhancing BPEL with Human Activities

Human Activities as part of the business process model are of particular interest to TAS$^3$. This is, because there is an explicit representation of human interactions with the process and the possibility to specify handover of control from the BPM system to persons and vice versa.

BPEL4People [12] and WS-HumanTask [13] are two standardisation proposals, complementing each other, intended to integrate human activities more closely into BPEL processes. A human activity is an activity which is executed by a human and is modelled explicitly as an activity in the business process model. It is modelled as a ”black box” from the perspective of the business process management system, with input/output parameters. I.e., when a call activates the human activity, the business process management delivers the data necessary and waits until completion to get the control back. BPEL4People provides five basic states of a human activity: running, completed, failed, terminated, and obsolete. A common way to implement a human activity is to define a web form which lets the user receive and deliver the parameters. The control goes from the business process management system to the individual who possesses the role, which is related to the human activity at execution time. While WS-Human Task standardises interfaces to human tasks, BPEL4People standardises human tasks themselves, e.g., by explicitly modelling the implementation of human tasks as web forms, by managing task lists for users, who are stakeholders of the human tasks, and by carrying out the role assignments for these tasks.

WS-HumanTask defines several generic human roles with respect to tasks (and notifications), i.e., ways how individuals are connected to tasks. There is a task initiator who may be undefined. Task stakeholders are ”ultimately responsible for oversight and outcome” of the task, but at least one person must be assigned to this role. Potential owners receive the task so they can take it over and complete it. The present owner is actually performing the task. The excluded owners may not become actual or potential owner. Finally, there are the roles business administrators and notification recipients. People are assigned to these roles through logical people groups, literals or expressions. Logical people groups are bound to a so-called people query specifying the membership of people in the group at deployment time and evaluating it at task creation time, possibly taking parameters into account. The format or evaluation mechanism of people queries is beyond the scope of the specification, i.e., there are no restrictions. BPEL4People allows defining start and completion deadlines. When a deadline is missed, an escalation is triggered, resulting in a notification or a reassignment of the task. Further, the standards enables to delegate and forward tasks but without affecting security rules. Potential owners, actual owners and business administrators can delegate a task. The delegate then becomes the actual owner. Forwarding a task replaces the forwarder with the recipients as potential owners.

As shown, BPEL4People and WS-HumanTask only deal with tasks and permissions to execute tasks (or delegate them, etc.). They do not address permissions on other resources such as, say, data processed, which actors in business processes might need. As the process can set the generic human roles for each task as literals, it is possible to implement arbitrary access control policies (including assignment of actors and separation of duty in particular) manually within the process model. People queries are a suitable mechanism to interface with an external access control component. However, WS-HumanTask and BPEL4People do not specify any particular authorization model themselves: It is not possible to restrict the recipients of task delegation and forwarding. This is undesirable from a security perspective.

In TAS$^3$, human activities are a way to explicitly integrate user interactions via web user interfaces into the business process. With this, business processes not only interact with web services (e.g. providing access to data as personally identifiable information) or other business processes (e.g. running at a service provider side), but also with users via web interfaces. These users are subjects of authorization, and BPEL4People allows user abstractions in the process model with roles. This is a prerequisite to effective role-based access control (RBAC).

## 2.2 The Intalio System

The Intalio System [14], an open source software for business process management with its Business Process Management Suite (BPMS) [10], comprises several components:

- A graphical BPMN modeller, the Intalio—Designer [15] with BPEL generator.
- The business process management system Intalio—Server [16] with the open source component Ode [17] as BPEL execution engine.
- Tempo [18], which provides a (not yet comprehensive) implementation of the BPEL4People specification, in order to support human activities in business processes and the role management for the individuals.

In TAS$^3$, the Intalio System serves as technical business process management framework. There are several reasons for this choice. Core parts of the system are open source software. So the integration of the planned enhancements of the modelling and execution components into the BPM framework is feasible. Besides that, Intalio already puts much emphasis on modelling and execution support for human tasks and role management in business processes, which are relevant business-level prerequisites of security aspects of processes. Finally, there exists a complete chain of strongly interacting components, from modelling at the business level with BPMN over transformation to BPEL as standard execution language of business processes in a service-oriented environment down to the execution engine for web service based business processes.

### 2.2.1 Intalio—Designer

Intalio—Designer (see [15]) is a modelling tool that supports the BPMN notation and can generate executable BPEL process definitions from business-process diagrams and additional information provided by the user, such as data assignments. Intalio—Designer also supports workflow-user interactions through integration with the Tempo project, which is an implementation of BPEL4People architecture, but still lacks compliance with the BPEL4People specification.

Security aspects are out of scope of the BPMN specification. While it is possible to model different participants in separate pools or lanes, the meaning of such a separation is vague and is not specifying security issues.

Intalio—Designer currently supports annotating pools representing individuals with organizational roles to define the authorization between processes and tasks of individuals. Beyond this, Intalio—Designer does not natively support modelling security aspects.

It is desirable to extend support for security aspects through annotations to facilitate specifying security policies declaratively instead of leaving these to be implemented in an imperative manner by business people or process modelling experts.

### 2.2.2 Apache Ode and Intalio—BPMS

Apache Ode (see [17]) is an open-source execution engine for business processes that follows the BPEL 2.0 specification and is governed by the Apache Software Foundation. It is used within the Intalio—BPMS [10] product suite to execute the BPEL processes and is integrated to provide seamless deployment from Intalio—Designer.

There are significant challenges related to the use of BPEL technology in secure distributed computer systems and together with web services. Of particular interest to the TAS$^3$ project are the problems of authorizing users to execute tasks within a workflow while enforcing constraints such as separation of duty on the execution of those tasks. Conducting end-to-end secure transactions between multiple participants is a further issue.

While BPEL may be coupled with a web-service-security infrastructure (i.e., WS-Security [19]) to provide integrity and confidentiality at the transport level, the BPEL language does not provide any support

for the specification of authorization policies or constraints on the execution of activities composing a business process. Hence, it is important to couple BPEL with a model to express such authorization policies and constraints as well as with a mechanism to enforce them. Further, it is important that such an authorization model be high-level and be expressed in terms of entities that are relevant from a modelling and organizational perspective. In the next chapter we illustrate this vision in more detail and with several examples.

A partial solution to these issues is the BPEL4People extension dealing with workflow tasks that require human involvement. A further solution, going beyond human activities, would be to manage security-related authentication and policy information using existing BPEL primitives. This could be achieved through explicit data assignments, explicit conditional logic, explicit invocations of security infrastructure services, etc. While feasible, this solution is not satisfactory because it clutters the process definition with imperative statements that could in principle be specified in a declarative fashion. Further, this would reduce one of the main assets of BPEL, namely to separate process logic from other concerns in order to achieve clearer and simpler process models.

Thus, if BPEL processes are to conduct secure transactions that do not only span individuals but also web services, a security model is required to be integrated with BPEL that is more general than the one offered by BPEL4People.

### 2.2.3 Intalio—Tempo: BPEL4People Workflow Model

Tempo (see [18]) is an open-source project to provide a workflow implementation as defined by the BPEL4People specification. Tempo supports role-based interaction of individuals and provides means of assigning users to roles. Further, it can delegate ownership of a task to a specific person and supports use cases such as escalation of responsibilities for tasks (ownership) and assigning individuals to roles. The Tempo project has started shortly after publication of the original BPEL4People whitepaper (see [12]), but before the first BPEL4People specification draft has become available. Due to this parallel evolution, there are some differences between the two. Here is a high-level summary of the differences:

- Tempo's design was based on what is referred to as "Constellation 4" in the BPEL4People specification. In other words, it relies on the standard BPEL `<bpel:invoke>` activity, i.e., the call to any activity as e.g. web services, instead of the ¡b4p:peopleActivity¿, that establishes a call of a special human activity. Therefore it implements its own protocol between the process engine and the task-management services (i.e., WS-HumanTask implementation);
- Tempo uses standard BPEL data-assignment activities to manipulate individuals and role assignments instead of extending the ¡bpel:assign¿ activity. So role assignments are not handled as first class citizens;
- Tempo does not use process-related human roles such as the `<b4p:processInitiator>`, `<b4p:processStakeHolders>` and `<b4p:businessAdministrators>`, which are roles for executing (meta) process management tasks ;

In addition to standard BPEL4People capabilities, Tempo provides:

- A basic role-based access control (RBAC) approach that integrates with most Lightweight Directory Access Protocol (LDAP)-compliant directory servers;
- Integration with various web-based user-interface technologies such as XForms [20], Intalio—RIA, and the TIBCO General Interface [21];
- A user-friendly task list for workflow participants that displays outstanding tasks and notifications and allows the kick-start of processes using people-initiating forms realising the user client and task list component of the WFMC.

# 3 Scenario and Requirements Analysis

In this chapter, we will derive and illustrate requirements using a business application, namely Accreditation of Prior Learning (APL), which is in production use at Kenteq. It is realized in form of hard-coded applications with additional, manually performed steps. To do so, we will first shortly present the methodology we used to model this application as a business process, and the process model we have derived. Then we refine the requirements which are roughly set up in Deliverable D1.2 [5] and D1.4 [22], related to secure processes, adaptation of processes, semantics of processes, and to the TAS$^3$ architecture (see D2.1 [23]).

## 3.1 Modelling of an Example Process

### 3.1.1 Modelling Methodology

Business Processes have become very popular recently thanks to the adoption of key standards such as BPMN and due to the fact that technologies have matured. However it is important to keep in mind that Business Process Management is first a methodology that is enabled by technology and not the other way around. Intalio has provided training programs in modelling by process experts to various TAS$^3$ members in order to transfer knowledge on the fundamentals of process modelling. During this step, participants realized that BPMN is a very advanced language, and Intalio—BPMS can support most of the real-life in-company processes. On the other hand, one quickly observes limitations when dealing with inter-company processes where one has to share parts of his business processes. Security then becomes a major concern, and it is important to capture the security concerns at the process modelling stage.

Intalio has developed the Process Modelling Framework (PMF) (see also [24]), as collection of best practices for business-process modelling. Intalio is using this framework to model business processes for its customers.

Agile software development methods have influenced this approach. Further, it is process-oriented by letting the business experts build and test business rules and process logic. This means that they obtain a stable process model with the corresponding rules and metrics before any code is written.

Intalio has derived some ”golden rules” and suggests the following modelling procedure:

- A ”top-down” approach for process modelling for discovery and ”middle-out” implementation of business scenarios.
- (Re-)use of established process patterns, centralised business rules, and existing services.
- Room for change control in the process flow, e.g., by using an interface layer to communicate with services.
- Store data once and share it where needed.
- Write reproducible test cases to validate business scenarios.

The top-down modelling approach is achieved in line with the various layers of process diagrams, from one high-level diagram covering the overall process flow (the ”core” process) via phase-level diagrams to detailed scenario-level diagrams, possibly over several layers as well. Implementation then starts ”middle-out” from the scenario level: Each scenario-level diagram is implemented using implementation- and service-level processes to decouple interfaces and to prepare for possible changes. Then the next step designs ”bottom-up” the interfaces to pass data needed for process flow-control.

Each scenario should be self-contained and testable as a stand-alone mini-process. An administrator can change scenarios at run-time without affecting the rest of the process. (Of course, this is only possible as long as the interface does not change. The PMF approach makes it easier to keep interfaces stable by decoupling the technical implementation using the integration and service levels, and by persisting data outside of the process.)

The PMF approach is described in more detail in [24].

### 3.1.2 The ”Accreditation of Prior Learning” Process

Kenteq’s business includes accreditation of prior learning (APL). APL is the common name given to the process of recognising the competencies an individual has gained through formal and informal learning in various settings. Recognition in this case means awarding certificates or diplomas on the basis of a generally recognised standard, such as the qualification structure for vocational education. A thorough description can be found in Section 4.4 of Project Deliverable D9.1 [25].

This process helps demonstrating the TAS$^3$ capabilities as part of the employability pilot. The PMF approach as described in Section 3.1.1 has been used to model parts of the Kenteq APL business process.

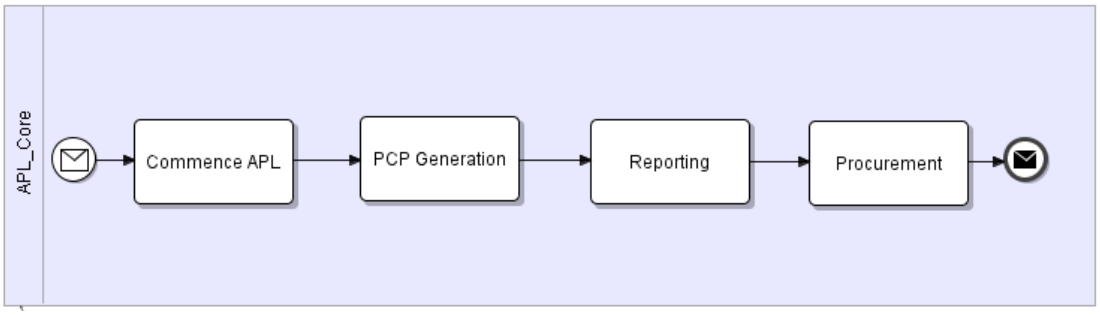


**Figure 3.1: The Core process diagram of the APL process**

Figure 3.1 gives a high-level overview of the different phases of the APL process, the so-called process core. This diagram is the first step for top-down process modelling and does not yet contain any implementation aspects. It consists of four high-level activities:

- The Commence APL phase contains administrative tasks needed to initiate the APL process.
- In the PCP Generation phase, the individual seeking APL (i.e., the candidate) and Kenteq employees work together to create and validate a personal competencies profile (PCP).
- Afterwards, the Reporting phase assesses the abilities of the candidate and creates a formal report.
- Finally, the Procurement phase encompasses administrative activities like ”mailing the report”, ”making out an invoice” and ”permanently storing the report”.

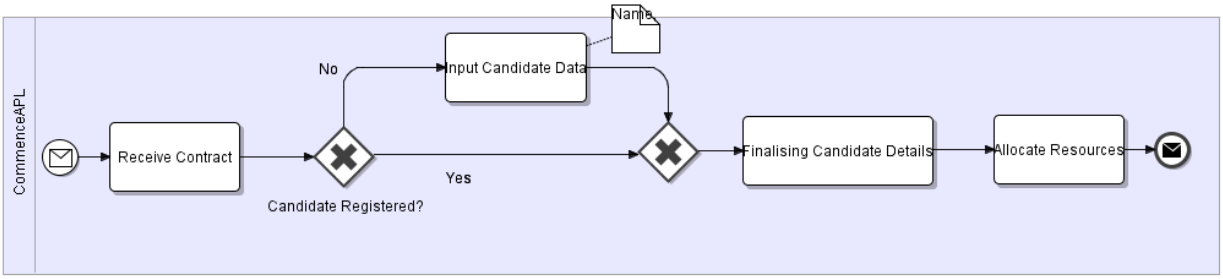


**Figure 3.2: BPMN diagram for the ”Commence APL” phase**

We have modelled three of these phases in more detail. APL is quite a sophisticated process; note that each activity of the phase-level diagrams has to be modelled in more detail as a scenario-level diagram.

From the ”Commence APL” phase:

- ”‘Receive Contract”: introduce the contract between the candidate and Kenteq as service provider of the APL application into the APL process.

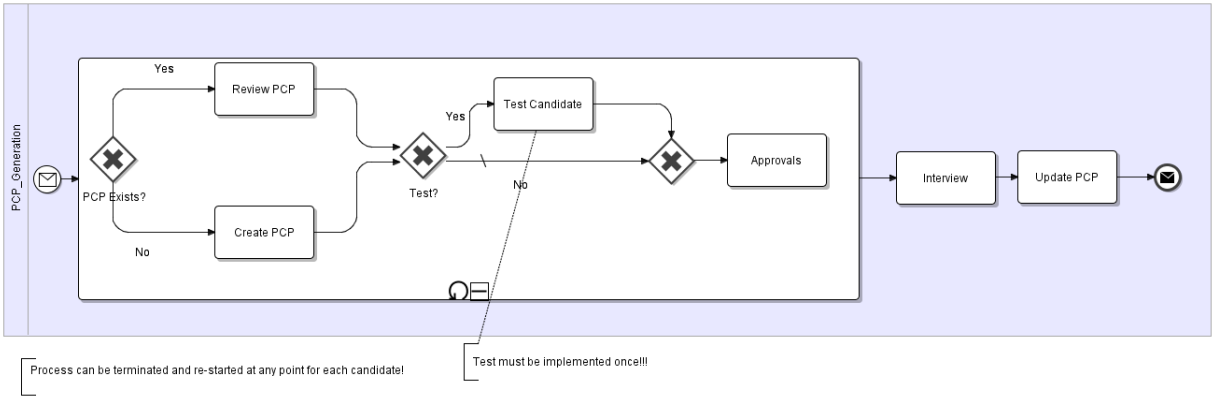


**Figure 3.3: BPMN diagram for the ”PCP Generation” phase to create or complement the personal competencies profile**

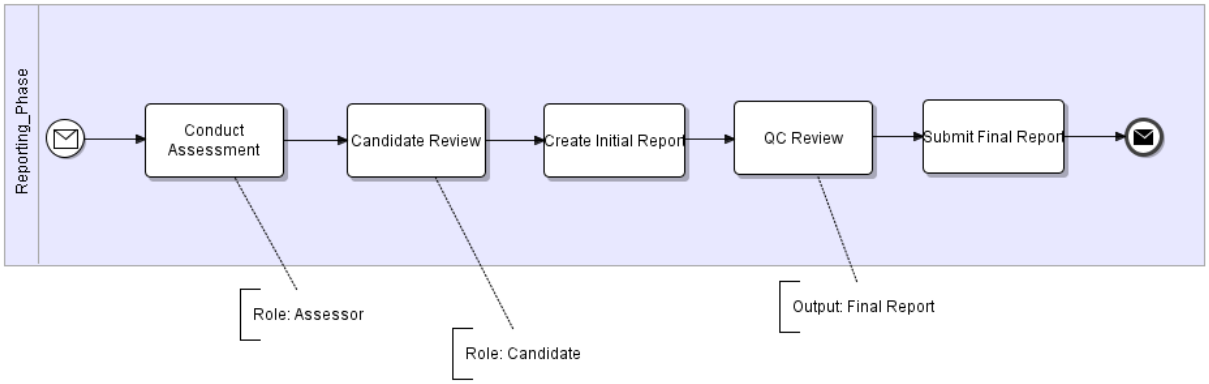


**Figure 3.4: BPMN diagram for the ”Reporting” phase**

- ”Input Candidate Data”: collect identifying data about the candidate only if he or she is not yet registered.
- ”Finalising Candidate Details”: check if the identifying data required is on-hand, otherwise complete it (manually).
- ”Allocate Ressources”: allocate the persons involved in the APL process, e.g. the coach, who will guide the candidate through the APL process.

From the ”PCP Generation” phase:

- ”Review PCP”:
- ”Approvals”
- ”Finalising Candidate Details”

The APL process involves several actors. This list is partly taken from Section 4.4.4 of Project Deliverable D9.1 [25]:

- The Candidate is an employee who applies for APL.
- The Organiser is a Kenteq employee who is charged with administrative tasks.
- The Coach helps the candidate to complete the portfolio and checks the evidence.
- The Assessor executes the APL procedure.

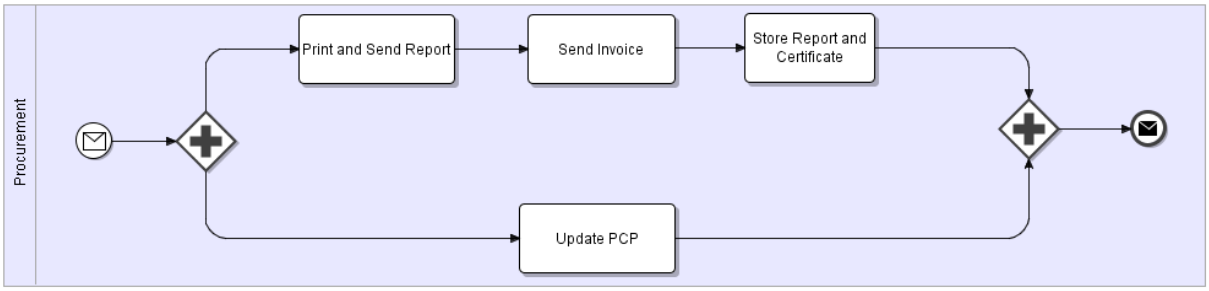


**Figure 3.5: BPMN diagram for the ”Procurement” phase**

- The Quality Controller ensures that the process meets the quality standards of the APL code and approves the quality of the report.

We will use these diagrams in the following to illustrate and refine the requirements we have identified.

## 3.2 Requirements

In this section, we will identify the requirements on a business process management platform. The presentation consists of three parts, matching the major research areas of WP3: Secure processes, secure process adaptation and support of semantics specifying security of business processes with ontologies. The APL process is a comprehensive legacy application, and we have not yet modelled it in full detail. Consequently, we expect to discover refined and new requirements when using the concepts described in this report for the implementation of APL as a TAS$^3$ application, as well as from the other pilot scenarios. Further, we are going to enforce the integrated treatment of secure processes, ontology and adaptability in the next phase. So we expect to identify requirements crossing the boundaries between these areas currently handled in isolation.

For Requirements Analysis we start with the Accreditation of prior learning (APL) scenario. The APL results in a business process within one company (Kenteq), with various manual activities. Deploying APL as a TAS$^3$ application means that personally identifiable information (PII) from other sources in the TAS$^3$ ecosystem will be made available to Kenteq, and PII collected at Kenteq will be sent to other service providers for processing. This means that stronger security is needed, but this must not impair the efficiency of the business process. [26] summarizes our findings of requirements on security for business processes and their impact on advanced security mechanisms.

### 3.2.1 Secure Processes

To facilitate secure processes, a comprehensive approach to security, spanning modelling, deployment and execution of processes, is necessary. We structure the requirements concerning the security of processes, assigning them to several topics: First, process modelling is a cross-cutting concern relevant for all concepts for business process security. Runtime security management and enforcement consists of two parts: on the one hand, authentication and identity management, and on the other hand, authorization. Finally, we identify requirements on the runtime behaviour of business processes. We shortly present each topic and list the requirements that belong to it, before presenting each one in detail. Note, that the names of the requirements start with the number of the deliverable, in which they are introduced.

1. BPMN modelling tools such as the Intalio—Designer supports graphical modelling of business processes (D1.2-R3.1). However, the models must be enhanced to address security issues. D3.1-R.1 calls for such enhancements in general, but specific implications on the modelling process are contained in several requirements, namely D1.2-R.3.5 ”task assignment” and D1.2-R.3.8 ”separation of duty”, and in the more detailed requirements presented in this report, namely D3.1-R.3 ”explicit assignment of resources, and of users to roles”, D3.1-R.4 ”role mapping”, D3.1-R.6 ”separation of duty”, D3.1-R.7 ”binding of duty”, and D3.1-R.8 ”specification which data process participants may access at what time”.

2. Authentication and identity management are of less importance to WP3. Our focus is on the compatibility of the workflow component with the authentication and identity management infrastructure provided by other work packages, as specified in D1.2-R.3.4 and D3.1-R.2.

3. The major focus of our secure business process management platform is on authorization, as specified by D1.2-R.3.5 through D1.2-R.3.8 and D1.2-R.3.10, and D3.1-R.3 through D3.1-R.8. The requirements aim at a flexible and powerful approach based on process roles and consideration of the context that business processes provide. All these requirements have implications on both modelling and runtime execution.

4. Finally, we are concerned with the runtime behaviour of business processes. D1.2-R.3.9/D.3.1-R.9 requires that processes can detect and recover from security violations. Requirement D3.1-R.10 calls for a possibility to specify under which identity, or even under which delegation tokens the process is supposed to act.

#### 3.2.1.1 Requirement D3.1-R.1: Visualization of security specifications at the process-design level

Several specifications exist today to address security concerns at the transport level (e.g., WS-Security/SAML [19]). However, currently there is no method to easily expose security concerns visually to the process analysts. We need to annotate or enhance BPMN as well as the modelling tool, in order to accomplish this.

This requirement is a refinement of D1.2-R3.1. As modelling crosscuts all security issues, it intersects with several security requirements, as already described above.

#### 3.2.1.2 Requirement D3.1-R.2: Distributed identity management

Actors in business processes need to be authenticated. This authentication must be based on a distributed identity management system (federated identity), because business processes in a service-oriented architecture comprise actors from different organisations. It is desirable that they can use a single account to participate in these processes. E.g., their employer or a national registry could issue their account.

The Kenteq APL process, for example, spans the employability agency, the company of the candidate and the coach and assessor, who might be independent consultants. They should be able to login using existing accounts. This implies that Kenteq needs to consult the security infrastructure to perform the authentication, and that there is a mapping between the account at Kenteq and the federated identity of the user. Further, Kenteq should be able to obtain information about them (e.g., special domain expertise) from external sources (e.g., a directory of employability professionals), also requiring such a mapping.

WP7 (Identity Management) deals with the distributed identity-management framework itself. The workflow management engine needs a component that can check the credentials presented. This is provided by the Credential Validation Service (CVS) described in D7.1 [27].

This requirement is D1.2-R.3.4.

#### 3.2.1.3 Requirement D3.1-R.3: Flexible user/role assignment

The choice of actors in workflows can be based on different application-dependent criteria. It should thus be possible to explicitly model the assignment of users to workflow tasks and workflow roles.

In the Kenteq APL process, this is necessary for the assignment of the coach and assessor roles: It is not possible to designate a number of Kenteq employees as coaches and just pick any of them, because special skills and qualifications (that depend on the APL candidate and are thus different for each process instance) need to be taken into account, either by a human or automatically by the process logic.

This requirement corresponds to parts of D1.2-R.3.5.

#### 3.2.1.4 Requirement D3.1-R.4: Role mapping

Some roles in business processes map directly to organisational roles that are independent from specific business processes and are managed in an independent repository, like an LDAP server. This is applicable to the Manager role in the Kenteq APL process: A manager at Kenteq can perform this function in any instance of the APL process as well as in other processes.

Accordingly, it should be possible to directly refer or map from process roles to organisational roles, without having to explicitly assign actors to such roles.

This requirement covers a very specific aspect of D1.2-R.3.5.

#### 3.2.1.5 Requirement D3.1-R.5: Least Privilege and Strict Least Privilege

In order to fulfil their tasks in a workflow, actors typically need access to certain relevant data. The principle of least privilege demands that a person should only get the permissions that he needs to fulfil his tasks. In our use case, an example of this principle is that a coach should only be able to access the profiles of candidates whom he is actually coaching (as opposed to those of all candidates doing APL at Kenteq). In other words, he cannot exert the coach role in unrelated cases. The principle of strict least privilege further demands that he holds these permissions only as long as needed to execute the task. For example, the candidate will be allowed to change his profile, but only at certain times (and not when it has already been reviewed by his coach or assessor).

To provide actors with the information they need to fulfil their tasks, it must be possible to specify the resources that actors of the process may access. Further, it must be possible to specify when access is allowed and when not (or only in a limited way, e.g., read-only).

This requirement corresponds to requirements D1.2-R.3.6 and D1.2-R.3.10.

#### 3.2.1.6 Requirement D3.1-R.6: Binding of Duty

Binding of duty is, in a sense, complementary to separation of duty described in requirement D3.1-R.8: Where separation of duty requires two activities to be performed by different persons, binding of duty requires them to be performed by the same person.

For example, the person that enters data and is asked to complete that data if it has been found incomplete by someone else should be the same person. In our example, Kenteq's APL business process, most roles (e.g., coach and assessor, and, of course, candidate), imply a binding of duty for all activities of that role.

Thus, it must be possible to impose binding of duty on roles.

The binding of duty requirements is derived from D1.2-R.3.5.

#### 3.2.1.7 Requirement D3.1-R.7: Delegation

It can become necessary for someone involved in a process to delegate this involvement - including all permissions - to someone else.

Example: Coach Bob goes on vacation for three weeks. He is involved in two running APL instances. The day before his vacation, he decides that Charlie and Dora would be best suited to jump in for him, and they agree to do so.

The delegation refers to Bob's involvement in two specific APL instances. It must be possible to delegate the ownership of process roles on the instance level. I.e., Bob delegates the first of the running APL instances to Charlie and the second one to Dora. This way of specifying the scope of a delegation is specific to business-process management, but is actually an example of the generic principle of constrained delegation of authority.

The permissions transferred by this delegation apply to tasks in the business process as well as to permissions on external data repositories. Thus, the handling of tasks must respect delegations. Furthermore, permissions that are assigned and activated in the context of a business process must also apply to dele-

gates. Finally it should not be possible to delegate role ownership to arbitrary persons, because this would compromise security. Instead, a delegation policy is necessary to determine which delegations are allowed and which are not.

This requirement corresponds to D1.2-R.3.7.

#### 3.2.1.8 Requirement D3.1-R.8: Separation of Duty

To avoid conflicts of interest (thus, ultimately for security purposes), it is sometimes required that several people cooperate to achieve a specific result (e.g., requiring two signatures of different persons to cash a cheque). This is usually done by requiring that several tasks that are all needed to achieve the desired outcome may not be performed by the same person. In any given instance of the Kenteq APL process, the same person may not act both as coach and as assessor. The reason for this is that there is a supervision relationship between coach and assessor. It must be possible to specify separation of duty constraints, to be enforced automatically at runtime.

This requirement corresponds to D1.2-R.3.8.

#### 3.2.1.9 Requirement D3.1-R.9: Recovery from security violations

TAS$^3$ as a trusted and secure architecture will perform various security checks at runtime. If security violations occur, they might be discovered. For example, when explicitly assigning users to roles, the organiser might nominate a clerk as coach, though he is not eligible for this role. Or he might nominate the same person as both assessor and coach, which is precluded by the separation of duties constraint mentioned above. When trying to access a resource, access might be rejected. E.g., consider the case that a candidate's PCP has been imported from an external source. It contains references to diplomas of the candidate stored in an external repository. The process requests a diploma on behalf of the coach for inspection, but the request is rejected, say, because of misconfigured policies or because of an explicit setting from the candidate. Processes must be able to react and recover when activities fail due to security violations, for example by initiating a break-the-glass procedure (cf. Chapter 3 of Project Deliverable D7.1 [27]). These recovery mechanisms must be specifiable in a flexible and systematic way, in order to be able to react to unforeseen security violations and to keep process definitions concise.

Security violations can happen in all processes, e.g., because of permissions that have changed. In the Kenteq APL process, this might be the case for operations on the candidate's portfolio.

This requirement corresponds to D1.2-R.3.9.

#### 3.2.1.10 Requirement D3.1-R.10: Acting on behalf of other entities

One purpose of business processes is to coordinate the fulfilment of tasks by humans. However, processes also orchestrate automated systems running autonomously. They call services, and they frequently do it on behalf of or in the interest of a human actor involved in the business process. When, for example, the process appends an approval certificate to the PCP of the candidate, this is done on behalf of the assessor, who has explicitly approved the PCP previously. Thus, it must be possible to specify for whom a process is acting. To ease maintainability, this should be possible in the business-process model.

### 3.2.2 Secure Process Adaptation

In order to enhance the adaptability of business processes (workflows), existing solutions, mostly supporting static adaptations of process models, are not sufficient in TAS$^3$ applications. (These are described in Deliverable D1.1 [28] which contains an overview of the state of the art.) Basic support for static adaptations is already contained in existing BPM systems. But there also is significant need for further research into dynamic adaptations, e.g., in order to facilitate more user support or to support cooperation of business processes that are defined by a business-process choreography (see e.g., [29], [30]).

In TAS$^3$, we need to improve the flexibility of processes also taking in account the context of running instances, e.g., users involved and their security and trust preferences and settings. In business process models there already exists the possibility to statically model alternative sequences of activity flows, i.e., a fixed defined process model supports well-known (but not too many) variations of the process. What we

want to achieve with dynamic adaptability goes way beyond this possibility.

We distinguish the following levels of flexibility as process categories:

- Ad-hoc-workflows: without completely specified process schema.
- Workflows in an open dynamic environment: need to be dynamically adaptable in varying the invocation of particular web services or other processes, or even changing the specified sequence of activities, when these activities are not known at process creation time. This may be required by the context, e.g. the actor of the process or the data involved, or by a changing security environment.
- Production workflows, not very flexible but robust. These are the static adaptations that are possible today.

In TAS$^3$ we primarily aim to support workflows in an open dynamic environment. In very particular situations even ad-hoc-workflows could be desirable, e.g. if there is the need to define a business process accessing data required for specific users, when neither the users nor the data are known beforehand. If there exists a great variety of data and of ways to access this data, it may not be feasible to design all possible sequences of tasks, decision points and branches of subworkflows in advance, and the BP manager may need to leave it to execution time to compose an adequate sequence of tasks.

Dynamic adaptation of processes can comprise the following functionality:

- Change process schema. Note: In the following we sometimes use ”process model” as a synonym of ”process schema”.
- Perform basic structural changes, e.g., replace tasks, delete tasks, insert tasks.
- Change the process flow, e.g., alter branch conditions, use different data, insert human tasks, alter process flows depending on the requirements of the active role or on the current status of the process data.
- Apply adaptations to process instances, i.e., migration.
- Perform adaptations automatically, semi-automatically or manually.

To achieve adaptability of active process instances, the explicit modelling of adaptations is required in order to formalize the process status and to define well-formed processes, i.e. processes with a consistent structure with respect to business process execution requirements (e.g., so-called sound processes). Hence, we need adaptation operators describing the change of the process schema. Additionally, we must have means to migrate process instances to a process model that has changed.

The definition of security for business processes depends on the process model itself, on the web services referred to in the process model, on interactions with humans, i.e., human tasks, and on the data used during the process, i.e., the workflow data. These elements also are the building blocks of a business process. Therefore, adaptation of the process, i.e. change of these elements and their composition (as workflow) has an impact on the security state of the process, and we need to be able to identify the effects of adaptations on the security specification of the process and manage it.

Real world processes like the Kenteq APL process, see Subsection 3.1.2, are applications that are commercially used. Because up to now there has been a lack of advanced adaptation support for business processes, actual implementations use workarounds or even run with restricted functionality compared to their potential realizations using the adaptation support envisioned. So we have to analyse the requirements for adaptability by having in mind the TAS$^3$ framework envisaged and the enhanced possibilities for business processes in such an environment. As important categories of adaptation requirements, we have already identified runtime changes of workflow types and instances, changes initiated by stakeholders of the business process, user support for controlled workflow adaptations and adaptations resulting from changes of the data involved.

The respective requirements identified in Deliverable D1.2 [5] are D1.2-3.12 through D1.2-3.15, and they result from the following observations:

- Service providers will outsource parts of their business process to other service providers. To be able to do so, they must have sufficient information on the available processes (interfaces, assumptions, i.e. pre- and postconditions and effects, interaction behaviour, nonfunctional properties).
- Users expect to know as early as possible what PII they need to provide so that a particular business process can complete successfully, or to put it another way, if the process can complete successfully with the PII they are willing to contribute.
- Process flows are not always modelled in a fixed manner. Sometimes it is not possible to foresee all possible flows that may occur. In the APL context for instance, depending on the candidate, the process to perform the assessment or to choose an adequate coach may differ from the predefined way. Another example is the change of data that results from calling a subprocess or web service. In these cases adaptation of the active process is needed.
- Adaption of a process must result in a process that guarantees the same level of security.

There are further requirements regarding the implementation of adaptations. To achieve a better automation and/or user support, semantic information on services and processes is needed. To accomplish this, we need adequate ontologies of the respective application area and of the security concepts.

In TAS$^3$, the trust and security negotiator is involved in choosing web services with adequate security guarantees. The result is a web service that is adequate from the trust and security perspective. But the insertion of new web services in a business process typically is subject to restrictions. The substitution of single web services by other web services with the same interface is a basic feature of a service-oriented architecture, but only when using abstract web services in the process model. Otherwise, the process has to be stopped, remodelled, deployed anew, and the server has to be set up. If we wanted to provide more flexibility, e.g., to admit web services with slightly different interfaces or even subprocesses that are more adequate to fulfil the required trust and security conditions, our system should support more advanced adaptations of the process. Advanced categories of adaptations need new mechanisms and concepts considering the actual process status and checking if process modifications are allowed at all.

### 3.2.3 Semantics of Secure Business Processes

TAS$^3$ aims at developing a Trust Network (TN) where parties (e.g. Service Providers (SPs) and users) can interact with each other securely. Thus, we need to provide semantics to services, processes and data to enable interoperability on the TN. In the Semantic Web, ontologies define concepts for a domain in terms of their attributes and relations among these concepts in machine and human readable format.

Based on the requirements, we can identify several topics to be covered by ontologies. Firstly, the TAS$^3$ architecture must support different web services to access applications over a network, such as the TAS$^3$ TN, and execute these applications on their respective hosting servers (requirements D1.2-2.3, D1.2-2.4 and D1.2-3.12). Thus, service providers should provide a description of their available web services. The World Wide Web Consortium (W3C) has developed a syntax, called the Service Description Language (WSDL) [11], based on XML, to define the abstract operations and messages provided by a web service. Although it supports basic interoperability, it lacks proper semantics enabling the automatic discovery, composition and invocation of web services. For example, a human user would have to manually search through existing web services to combine them in a useful manner. Several ontologies (e.g. OWL-S [31] and WSMO [32]) are available to provide semantic mark-ups. Their use would allow automatic translation of message content between heterogeneous interoperating services. [33] provides an extensive comparison between WSMO and OWL-S. However, in TAS$^3$, we are only concentrating on interoperability based on security, privacy and trust.

Secondly, business processes define the order of tasks (also called sub-processes) to achieve a certain goal. One of the aims of WP3 is to enhance flexible adaptability of business processes (see Section 3.2.2). As a result, we need to represent business processes in terms of pre- and post-conditions of their sub-processes, the relationships between these sub-processes and the role of its participants. Moreover, the system must be able to enforce the separation of duty constraints (Req. 1.2-3.5 and 1.2-3.6 in Deliverable

D1.2 [5]). As a result, we need to extend the ontology with application-specific roles using these concepts to define business rules.

Thirdly, the TN should enable secure data interoperability between web services and business processes. Therefore, we need to develop an ontology describing the security-related aspects of data available within the system. For example, personally identifiable information (PII) should be specified in such a way that it is understood by all parties as sensitive data (Req. 3.9). As a result, the understanding of business processes (and of web services) is increased through the use of semanticallyannotated data.

Finally, parties on the TN should be able to set policies (authorisation, authentication, and trust) that control access to data (Req. 7.5, 7.6 and 7.7). Furthermore, the system should be able to detect (and resolve) conflicts when multiple authorisation policies have been defined (Req. 7.7). As a result, we need to describe the security mechanisms, e.g. Single Sign-On, used by a service and/or process to access assets.

### 3.2.4 Requirements on the architecture

TAS$^3$ aims at developing a Trust Network (TN) where parties (e.g. Service Providers (SPs) and users) can interact with each other securely. Business processes provide a means to explicitly handle interactions of services, processes and data at a modelling level and during execution in the TN. Especially, security comes into play for business processes, when they interact with the environment via web services.

In TAS$^3$, every service request as well as every service reply has to pass through policy-enforcement points (PEPs) on both the requester and the provider side. This is depicted in Figure 3.6 (taken from paragraph 1.6.2 of D2.1 [23]). Executable business processes can act both as a service requester and a service responder. They also interact with humans, either via front-end services as shown in the diagram or via a dedicated task-list console. The latter enables a type of relationship between business processes and humans not depicted here: The process acts as a service requester, creating tasks for humans to complete, and the human (through the task-list console as a proxy) acts as a service provider.

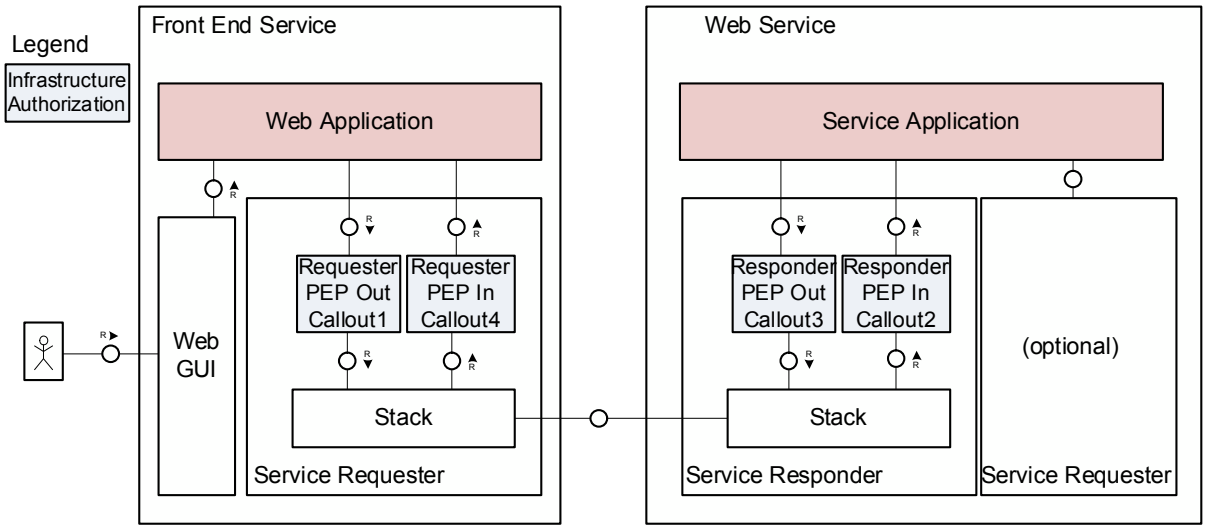


**Figure 3.6: A web service call passing through policy enforcement points**

An important aspect of security is authorisation (as shown in Figure 3.7, taken from paragraph 1.6.3 of D2.1 [23]). Authorisation is not static, but instead depends on the context and on the current state of an interaction. For example, the coach and the assessor only need access to the candidate's portfolio until they have completed their reports.

As business processes orchestrate interactions, this state becomes explicit and can be used as a parameter or context information to specify authorisation in a generic way, i.e. when checking the authorisation the actual context is available instantiating the authorisation rules.

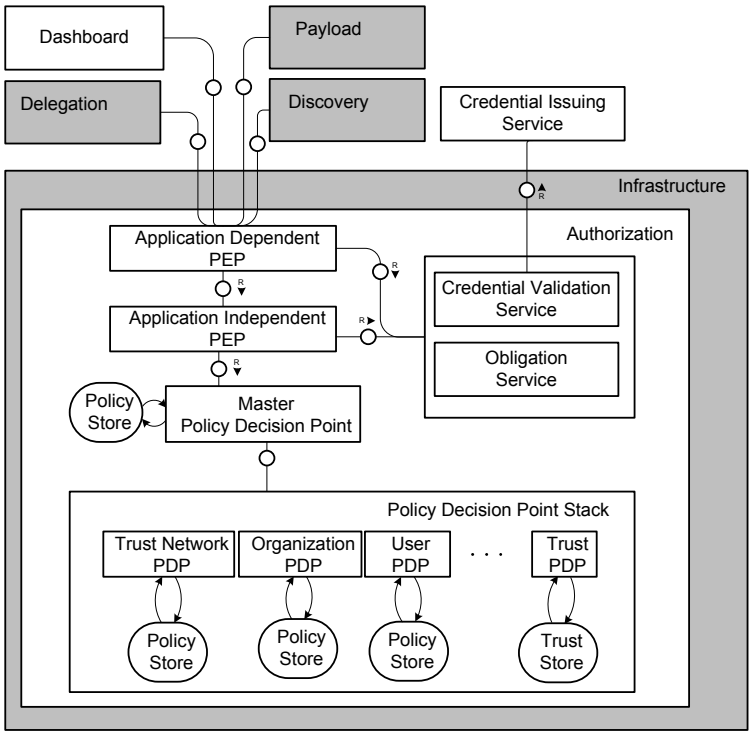


**Figure 3.7: Authorization in the TAS$^3$ architecture**

A security architecture with native support for business processes must be able to perform the following tasks at the enforcement level:

- It must maintain the state and the security context of each process instance. This includes the interaction partners (humans or computer systems) assigned to the process instance, the activation status of permissions that are only valid in the context of the process instance, and information necessary to enforce separation of duties.
- It must store and make available the security specifications for business processes (i.e., conditions for the assignment of roles, role delegation policies, separation of duty constraints, and specifications for the assignment of permissions to role owners). These policies apply to business-process models (types), not to individual instances. Thus, the specifications are generic, i.e., apply to all instances of a particular model. First, they contain an enumeration of the human roles, the web-service-interaction partners and the resources involved in the process. Second, they contain criteria which these entities must fulfil and, according to which, automatic selection of the entities possibly occurs.
- It must take the policies and state and security context of the process instance into account when checking explicit assignments of human actors, web-service-interaction partners, or resources, or when performing automatic assignments.
- It must allow processes to act on behalf of the actors involved, i.e. passing on such an authorization to services invoked, and it must allow the processes to pass on such an authorization to other human actors involved in the process. This can happen in several forms, e.g., through authorization tokens or through explicit calls of a delegation service. It must provide a central place for users where they can see which processes they are involved in, and where they can delegate their role ownership in specific business process instances.

Further, the identity of actors in business processes has to be aligned with the TAS$^3$ identity management infrastructure. The propagation of identity handles is described in Chapter 4.2 of [23]. When a user logs

into a frontend service, an identity provider authenticates his credentials and asserts his identity to the service, issuing an identity token. Each service knows the user by a different persistent pseudonym. Thus, when the frontend service needs to call another service to request data on the user, it must first acquire an identity token that the other service can use. The identity mapper service will provide this exchange.

# 4 Conceptual Design

This chapter gives an overview of the conceptual design to facilitate secure adaptable business processes. Figure 4.1 shows the main issues regarding secure processes. Security comes into play at two levels: The definition of security settings accompanies the modelling of the process itself. Regarding execution of the process, the security settings are enforced, taking the execution context into account. This work closes some of the gaps we have identified in [26]. Ontologies will provide semantic interoperability for security definitions. Ontologies provide support to ensure semantic correctness of the adaptation and a sufficient security level of the process adapted. Adaptation concerns both process models and running process instances.

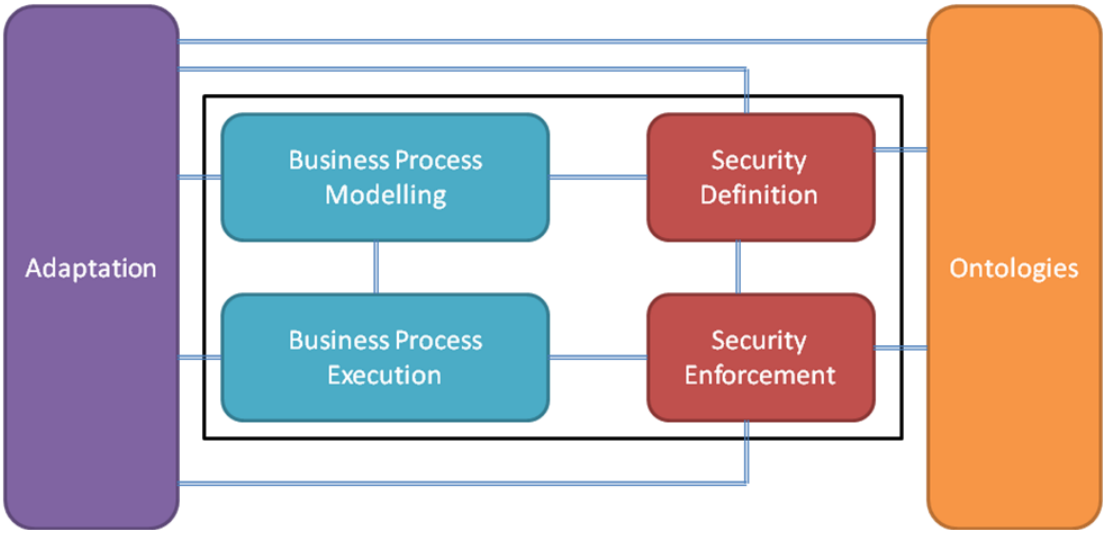


**Figure 4.1: Overview of the topics of security management in processes.**

## 4.1 Mapping to the TAS$^3$ Architecture

This section yields to present how the business-process-specific security management and the resulting components relate to the general security architectural framework of TAS$^3$. First, we begin with a description of the general security enforcement approach in TAS$^3$. Then we identify four categories of specific components to establish security for business processes. The rest of the chapter then presents for each category the requirements to components of this category and describes their functionality.

### 4.1.1 Security Enforcement in TAS$^3$

In TAS$^3$, any communication is subject to specified security and trust policies. Compliance is checked for every request and every reply, both at the service requester and at the service provider side as shown in Figure 3.6. This model is generic and is not specific to business processes, but on general service-oriented architectures. The book [34] gives also a good overview about existing security mechanisms for web services and service-oriented architectures, which will be enhanced and improved in this project. The policy enforcement points shown there (PEP Out and PEP In) will, accordingly, intercept any web-service call to or from the business process and enforce any applicable policy. Again, these policies are in general not specific to business processes, except that they can refer to properties of the process model or the process instance in question. Such properties may be the execution status of the process instance (such as activities waiting for execution, values of internal variables or the execution history), the security context of the process instance, the roles and resources assigned to the process, or the description of the process model, e.g., its privacy policy.

On the other hand, activities in processes can explicitly cause modifications of their security context, e.g., assign users to a process role. These modifications need to adhere to policies. Otherwise, users could

illegally go beyond their privileges. The business-process-specific security components as described in the next section will both. On one hand, they want to support the generic policy enforcement infrastructure by providing attributes necessary to evaluate policies. On the other hand, they will evaluate and enforce the process-specific policies.

### 4.1.2 Business-process-specific security components

We can broadly categorise the tasks of the components needed to establish security for business processes in the TAS$^3$ context into the following categories:

- Capturing and storing security-relevant information on instances of business processes.
- Runtime enforcement of security policies by inspecting incoming and outgoing messages.
- Management of configuration changes in other parts of the TAS$^3$ infrastructure.
- Creation of security configurations based on process models.

Figure 4.2 shows an architecture overview of the TAS$^3$ Framework in a service-oriented architecture from Deliverable D2.1 [23]. The business process (BP) box and the blue box enhance the architecture for business process handling. The blue box represents the security-related execution state of the business process, more precisely components that manage the business-process related security, based on process instance information about security. It shows how the business processes are integrated into the runtime and enforcement domain of the architecture. A business process uses the PDP to ensure authorized execution of process activities, like web service calls, subworkflow calls, use of resources as web service calls but enhanced with state information. This results in additional or enhanced business-process specific security components, which are described in the following. The modelling and configuration domain also affects the PEP of business processes. It will allow to configure the construction of the PEP and the involved security components. For their specification we use a business-process-modelling tool, enhanced to support security annotations.

From the perspective of security enforcement, Figure 4.3 shows Figure 2.10 of the TAS$^3$ report [5] with the arrangement of enforcement points in web service call flow. Web Service calls are represented in Client or Service Applications, both may be realized as business processes. Again, the blue boxes represent the components managing the security state information of process instances which enhance the security handling of business processes.

In Section 3.2, we have identified requirements the security architecture must fulfill. In the following Section 4.1.3 we state briefly how to meet these requirements by adapting existing or introducing new components.

### 4.1.3 Components managing instance-specific security information

The following architectural requirements hold for components for business-process management managing instance-specific security information:

**Information of business-role assignments to individuals.** We need to store which users are currently assigned to a given business-process role for a given business-process instance. A PDP can use the current role assignment to make access control decisions, or to another component can reconfigure PDPs in the infrastructure. In this case, a policy-information point (PIP) will provide it to the PDP or other components. In the case of instance-specific business-process roles, the processes are defined locally and used within the scope of one business process instance. Thus, the PIP assures for its own that the assignment is valid.

**Policy decision.** The Business Process PDP that checks the business process-specific security constraints defined in the business process policy. For example it acts as an decision point for task assignments to users.

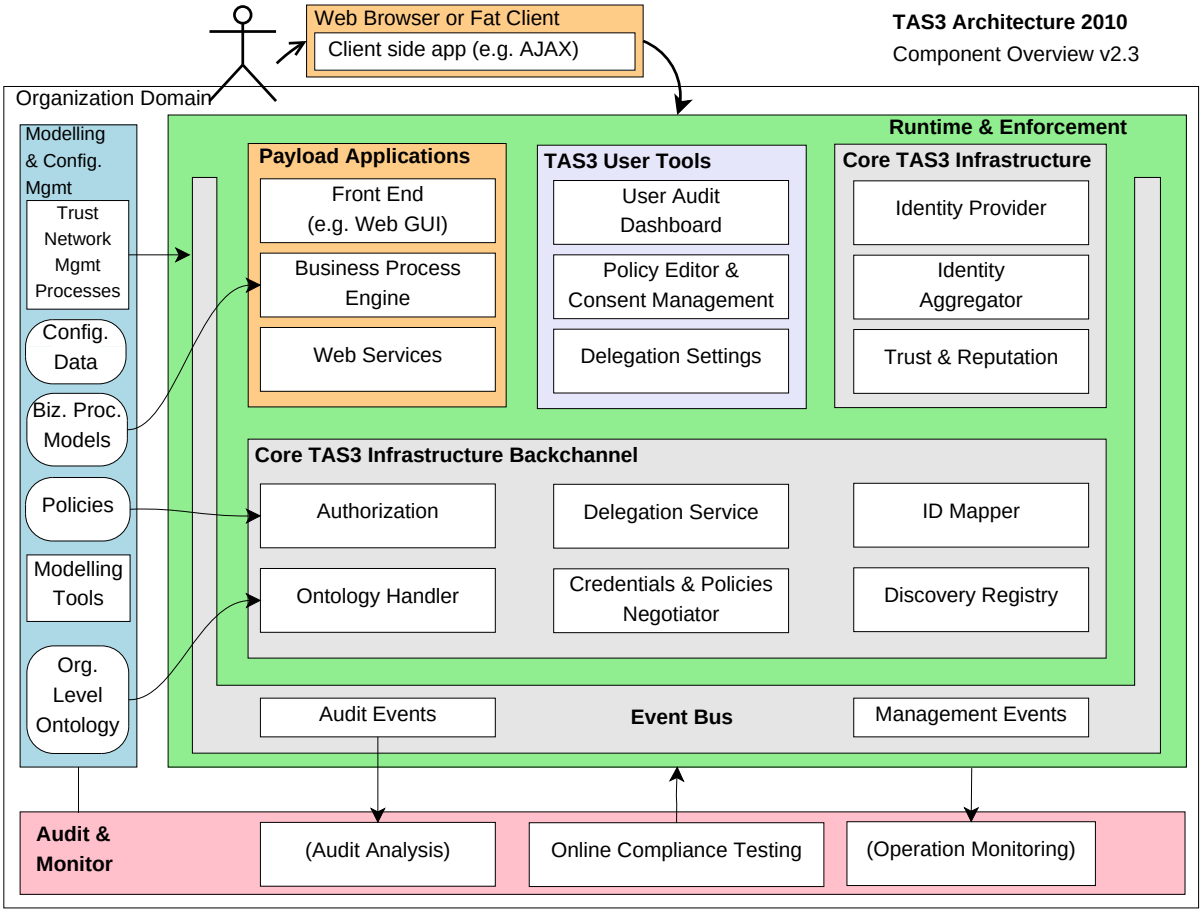


**Figure 4.2: Architectural Overview about Business Process Integration in the TAS$^3$ Architecture**

**Delegation handling.** When the process assigns a user to a group of tasks, this assignment must be valid, i.e., must conform to the business-process policy. If the role assignment is only used by the process management platform and not distributed across the infrastructure, it is sufficient to let a PDP authorize the role assignment and store it in the PIP. Otherwise, the business-process security components have to ensure that the user gets the permissions he needs to perform the tasks. To achieve this, they have to interact with the Delegation Service.

**Information of process-instance-specific attributes for policies.** Role-assignment and delegation policies can refer to attributes of the process instance that are specific to a certain business-process model. Accordingly, we need an **Instance-attribute PIP** that will make this data available to the PDP. Additionally, the business-process must be able to set these instance-specific attributes.

**Restriction of granting access rights to intervals.** To evaluate constraints that refer to intervals within a process, e.g. the time between start of task A and end of task B with A, B tasks in the process, the PDP must be able to determine whether the execution of a process instance is currently inside a certain interval. To this end, we will build an **Interval Monitor**. When a process instance starts execution, the Interval Monitor must determine the set of intervals, e.g. relevant for the delegation rules of the process model.

To store the instance-specific information we introduce the following component:

#### 4.1.3.1 Business Process Policy Information Point (T3-BP-PIP)

The Business Process PIP handles the assignments of process roles to individuals. For each process model, there exists a set of roles as well as assignment policies for these roles. The BP-PIP manages current assignments of individuals to roles, for each process instance. It checks requests to assign a

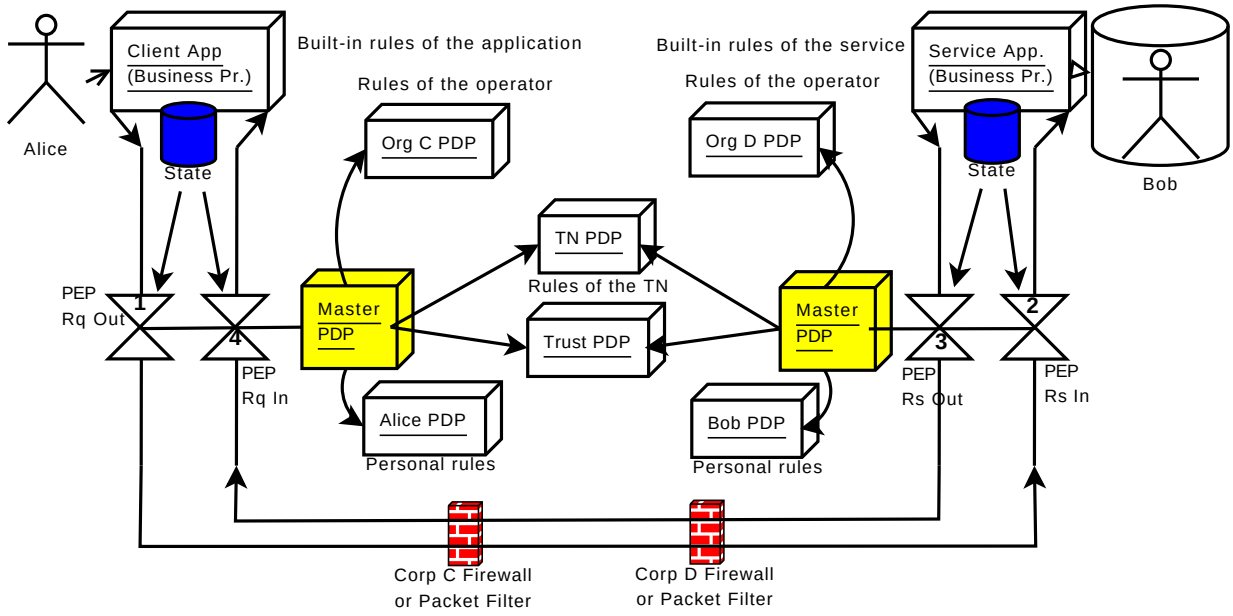


**Figure 4.3: Arrangement of enforcement points in web service call flow of business processes**

specific person to a role against the relevant assignment policy before it changes the assignment. Finally, it let other components query the current assignment.

The BP-PIP monitors the execution of process instances. It determines for each interval which is used for limiting the validity of a permission whether the process execution of a certain instance is currently within the interval.

Storing process values of process attributes in an instance-specific way is also a task of the Business Process PIP. For each process model, there exist the attribute names and the meaning of the attributes. The values are specific to instances and can be set by the process engine or the process instance at runtime.

### 4.1.4 Components enforcing security policies on messages exchanged

The following architectural requirements apply to components of business process management enforcing security policies on messages exchanged:

- The BPEL execution engine needs (application-dependent) policy-enforcement points for incoming and outgoing web-service calls (**PEP-in and PEP-out**). These PEPs determine any context information necessary to evaluate whether the call may be processed, in particular the ID of the business-process instance receiving or performing the call, and of the corresponding business-process model. It communicates the authorization decision to the BPEL execution engine.
- The BPEL execution engine needs to be enhanced so that it can throw BPEL faults when authorization is denied for outgoing web-service calls. It is also possible to handle such exceptions directly in the application business process.
- The authentication mechanism of the workflow component will be enhanced so that it uses single sign-on. Then, it can deal with the identity tokens returned by the identity provider.
- The Dashboard is the component of the security framework, that allows users to interact with the framework. It needs functionality so that users can see their current roles and delegate them to other current actors in the business process.
- Business process execution has to provide audit information for each security-relevant operation by pushing it to the audit bus. The dashboard can present this information in a user-centric manner.

- To meet confidentiality requirements all communication must be encrypted. The BPEL engine has to include an encryption layer implementing the WS-security standard. Parametes and keys for each web service call are determined dynamically using the TAS$^3$ security architecture.

Regarding WP3 these requirements are implemented by the following components:

#### 4.1.4.1 Service Requester Policy Enforcement Point (T3-PEP-BP)

The Business Process PEP groups a number of features:

- It makes XACML requests to a PDP to determine whether requests and responses passing through the PEP are authorized. This is the functionality commonly associated with a PEP.
- It logs the service requests and audit-relevant information to a specific component in the infrastructure (the Audit Bus).
- It looks up the endpoints to be used for service calls and routes request to the correct endpoint.
- It collects any attributes needed for making the XACML request to the PDP. They are available, inter alia, from the BP-PIP.
- It looks up the policies to be used for authorizing the web service request, if they are not already known to the PDP. Additionally, it performs necessary adaptions of policies to reflect instance-specific security requirements (policy template instantiation).

#### 4.1.4.2 Business Process Manager (T3-BP-CLIENT)

The Business Process Client makes sure that only authenticated and authorised individuals can access process instances. Authentication is accomplished by integrating single sign-on into the Intalio Tempo components. The Tempo components provide a graphical user interface for (human) tasks in business processes. Additionally, it determines the users allowed to perform tasks (based on the user-role assignment stored by the PIP and possibly the decision of the Business Process PDP).

### 4.1.5 Components managing the security configuration in the infrastructure

The following architectural requirements apply to components of business process management that manage the security configuration in the security infrastructure:

- For some of the resources delegated to the process will occur a further delegation to process participants, i.e. users holding roles in the process instance. We will provide a component, the **Business Process Delegation Service**, that will track the role assignments and assigned resources and issue delegations accordingly. These delegations will contain two kinds of restrictions, the second one being optional: First, the delegate still needs to hold the role in the process instance that has caused the delegation in the first place. Second, the execution state of the process instance must be within a given interval at the time the delegated permission is used.
- We need to store the policies and further security information related to certain business process models, namely role-assignment policies, role delegation policies, delegation rules, specification of instance attributes, and the list of resources. This information will mainly be stored in the components using it at runtime or in a general-purpose PDP. However, it might become necessary to use a more generic form. In general, such a generic form can not be directly used by a PDP. It must be pre-processed in order to be usable with respect to specific instances of business processes.

#### 4.1.5.1 Business Process Delegation Service (T3-BP-DEL)

The Business Process Delegation Service allows individuals to delegate their involvement in a specific process instance (i.e., the pair consisting of a process instance ID and a role) to another individual, who is then permitted to carry out the associated tasks, and is responsible to do so. Prior to the delegation, the DEL component will require consent of the delegate. Further, the delegation is only allowed if it adheres to a delegation policy under specific conditions. The policy must be specific to the process model, and must refer to attributes of the process instance, the delegator, and the delegate.

### 4.1.6 Components Creating Security Configuration

The following architectural requirements apply to components that allow to create a business process security configuration:

- User centricity is an important factor for defining business process security constraints. The configuration should take place in close contact to the business process model, directly within an additional layer to the graphical process model, at best.
- The security configuration has to be enforced on the execution layer. Likely the user-centric security configuration format has to be transformed to a format processable by the security enforcement components of the business process security architecture.

#### 4.1.6.1 Process Security Modelling and Configuration Component (T3-BP-SMC)

The Process Security Configuration Component allows to model security during the business-process-modelling task and related to the process model. To this end, the business-process modelling tool based on BPMN has to be enhanced by security-specification parts. Taking the security-annotated process model there will be a transformation of the security model to the enforcement and execution level.

#### 4.1.6.2 Business Process Administration Component (T3-BP-MGR)

The Business Process Manager is a tool of the business process framework to allow administration tasks regarding workflow adaptation. The T3-BP-MGR consists of two subcomponents, one handling the adaption of process models on the modelling layer, and the other for migrating active process instances on the execution layer.

### 4.1.7 Overview of the Architecture

Table 4.1 is a summary of the new, reused and enhanced components.

Figure 4.4 gives an overview of the components and their relations. The edges stand for possible communication between components.

- Process models are developed using a process modelling tool, e. g. Intalio Designer. Security aspects are extracted from these models using the security modelling and configuration component (T3-BP-SMC). Adaptation aspects are modelled and validated.
- Information on the business process security policy (e.g., assignments of tasks, binding of duty, etc.), are sent to the T3-PEP-BP, and information on the process model (task flow, adaptation configuration) are sent to the business process engine (T3-BP-ENGINE-ODE).
- The process engine (T3-BP-ENGINE-ODE) executes deployed process instances and calls payload web services or human tasks.
- When the process calls a (payload) web service, the request and response messages are passed through an policy enforcement point (T3-PEP-BP). The T3-PEP-BP examines the messages and

**Table 4.1: Overview of new security enforcement components**

| | |
|---|---|
| Business Process Modelling and Configuration Component (T3-BP-SMC) | Tool for configuring security constraint on business processes. |
| BPEL Execution Engine (T3-BP-ENGINE-ODE) *(existing)* | Executes business processes specified as WS-BPEL. |
| Business Process Client (T3-BP-CLIENT) | Controls everything relating to subjects interacting with the business process: authentication, role management, task assignment. |
| Business Process Policy Enforcement Point (T3-PEP-BP) | Monitors incoming and outgoing web-service calls and enforce decisions taken by the PDP. |
| Business Process Policy Information Point (T3-BP-PIP) | Stores instance-specific information required for all kind of security enforcement activities. |
| Business Process Delegation Service (T3-BP-DEL) | Handles the delegation of process tasks and resource access permissions. |
| Business Process Policy Decision Point (T3-PDP-BP) | Grants or denies authorisation according to business process policies. |
| Business Process Manager (T3-BP-MGR) | Tool of the business process framework for administration tasks regarding workflow adaptation. |

asks the T3-PDP-BP, whether the messages are authorized. This authorization contains necessary context, as determined by the T3-PEP-BP. If the messages are allowed, they are passed on to the process or its communication partner.

- If human tasks are called, the T3-BP-CLIENT and the T3-PDP-BP perform assignments of users to tasks. Non security relevant assignments are handled by the T3-BP-CLIENT directly, security relevant assignments are performed by the T3-PDP-BP. To achieve these security relevant assignments, the business process user client (T3-BP-CLIENT) makes a request to the T3-PDP-BP. All assignments are stored in the T3-BP-PIP.
- The T3-PDP-BP requests security policy information and potential business process users from a user database for the assignment of tasks to users.
- The T3-PDP-BP may apply process-specific policies (among others). In some cases, for example using sticky policies, the T3-PEP-BP passes the applicable policies to the T3-PDP-BP.
- If policies refer to process-specific attributes, the T3-PDP-BP requests and receives these attributes from the T3-BP-PIP.
- When permissions with interval constraints are used, the T3-PDP-BP contacts the T3-BP-PIP to determine whether the interval is currently active.
- The T3-BP-PIP retrieves events about running process instances from the process engine in order to determine the currently active intervals.
- A user's dashboard can request information about current delegation of access rights of the user and about roles held by that user.
- The business process manager (T3-BP-MGR) uses the adaptation configuration by performing adaptations caused by security constraints for running instances.

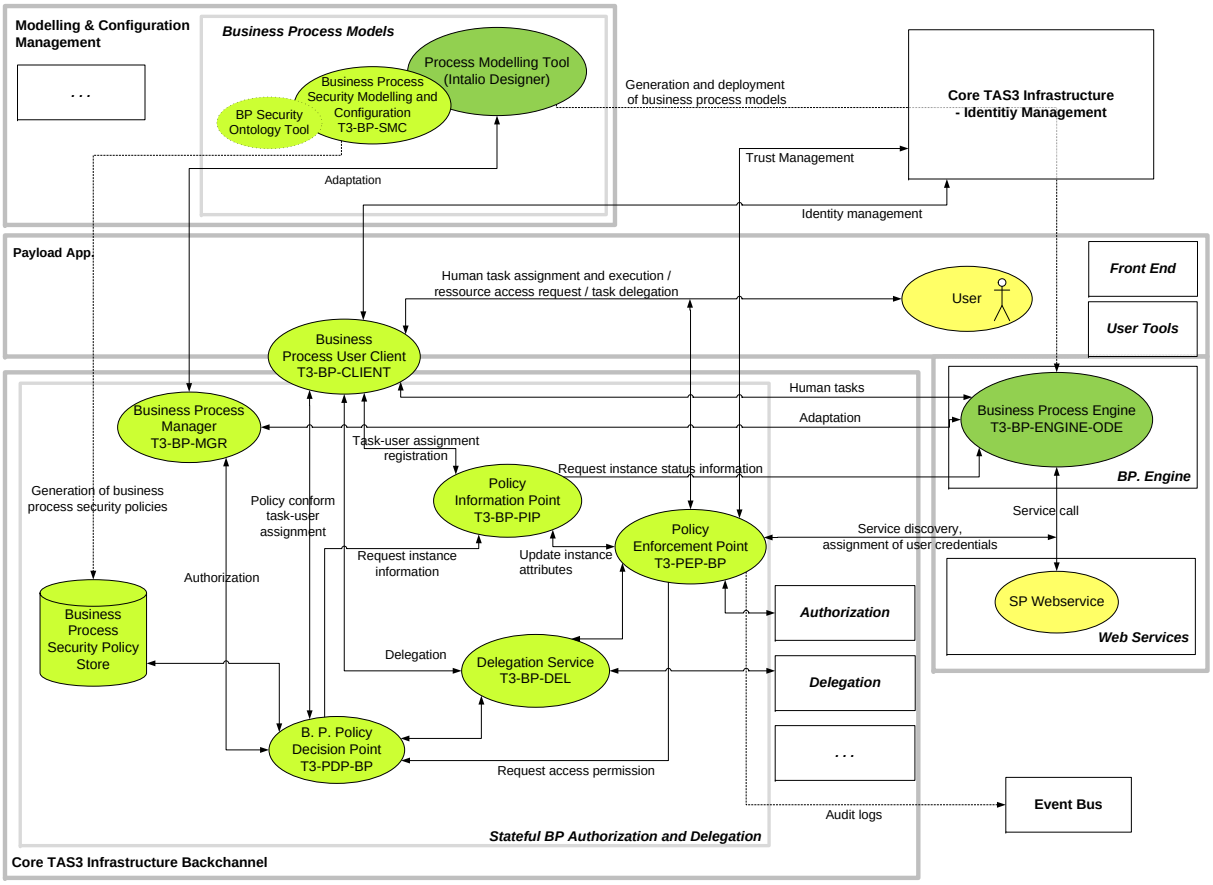


**Figure 4.4: Overview of the business-process-specific security components**

## 4.2 Using Business Process Modelling to Configure Security Components

### 4.2.1 Motivation and Overview

The TAS$^3$ architecture comprises a rich functionality, and some of this functionality needs to be configured carefully to ensure smooth operations from the perspective of the users. Users perceive such smooth operation as dependability and trustworthiness. It is a prerequisite for a Secure and Trusted Network to be accepted by the users.

Correct configuration will also be essential to ensure that business processes and services function correctly and securely. Given that most security technology is crucial and even tiny misconfigurations may lead to failure, it must be possible to correctly configure the trust network so that it will work properly right away. Our contribution to this reuqirement is to (1) descriptively specify security constraints, (2) automatically transform these constraints into enforcement level.

The modelling of the processes takes place at the business level. This is the right abstraction level to define security rules relevant to the business processes and their components. Therefore a model-driven approach seems useful to allow security specifications at the business level and transform them to the execution level, e.g. to security rules for processes as role definitions and delegations or authorization rules or other ways to configure the security framework.

The architecture document [4] lists a set of requirements which outputs should be possible to derive, such as:

- Derive parameters of the trust network level model to facilitate federation and single sign on con-

figuration, e.g., with white list of trusted parties or metadata for entities.

- Provide declarative statements on attributes needed by the clients as well as policies under which providers are willing to release attributes. This output can be used to automatically configure layers of Client Request PEP and Provider Request PEP.
- Provide policies, business process models, and interface descriptions as input for automated compliance validation.
- Derive security rules guiding selection of web services and use of secure entities (data), i.e. influence the discovery service for web services.

To accomplish this, we investigated in enhancing the business process modelling language by annotations of process diagram elements (see Section 4.2.2). [35] enhance UML models with security aspects, which shows some similarity to our approach, but we will use BPMN instead and have a more comprehensive language. In order to support the automatic configurations of security components with information from the business process modelling we developed a configuration component that uses input from an enhanced business process specification (schema) and transforms it into configuration parameters and descriptions for the security components, with respect to the requirements list given above. This approach follows a model-driven development. In [36] the architecture comprises a configuration component, which follows a similar approach for business processes. The main difference consists in the fact, that TAS$^3$ supports an open distributed trust network on which the business processes are running, and the fact that TAS$^3$ puts emphasis on user involved in the applications to be secured. To this end, the security framework, i.e. the target of the development, is more comprehensive. [37] proposes a model-driven approach to specify security with UML and generates a configured security infrastructure. We are starting from BPMN as process modelling language. In subsection 4.2.2 we present our approach to support security modelling in combination with the business process modelling.

Section 4.2.4 describes our approach to transform the security annotations of the business process model to the enforcement level. The upper part of Figure 4.4 gives an overview how a configuration and transformation tool works. Input is the business process description enhanced with security rules, like role information, authorizations for using business process elements, or auditing rules. The transformation and configuration tool transforms these business-specific security specifications into policies stored in the policy store and used from PDPs, and parameters to configure the policy enforcement point or the context store. Further, also parameters to configure the trust management can be derived, e.g., places in the process where users may have the opportunity to provide feedback about the behaviour of used components thus supplying the behavioural trust management. For more details, we refer to Section 4.2.4.

In Section 4.2.5 we describe an ontology-based approach for modelling security annotations of BPMN process diagrams. This contributes to improve usability for security modelling with supporting users, i.e. the process and security modellers, with a knowledge base of available annotations and its syntax and parameters even in a heterogenous environment with different vocabularies.

### 4.2.2 Modelling Security on the Business Process Modelling Level

Support for annotating BPMN process diagrams needs the following steps:

- list concepts of security which allow to define security goals,
- take a meta model of the BPMN which will be used for annotation, and
- determine concrete security annotations which will apply to business process elements.

For the first step we are taking a semantic description of goals for security modelling from [38]. Figure 4.5 gives a comprehensive overview about concepts for describing security goals. Some of these concepts are already part of the Project Trust Network, so we assume, for example, that communication over the internet will always need encryption. Others, especially authorization and authentication are more important security requirements for business processes.

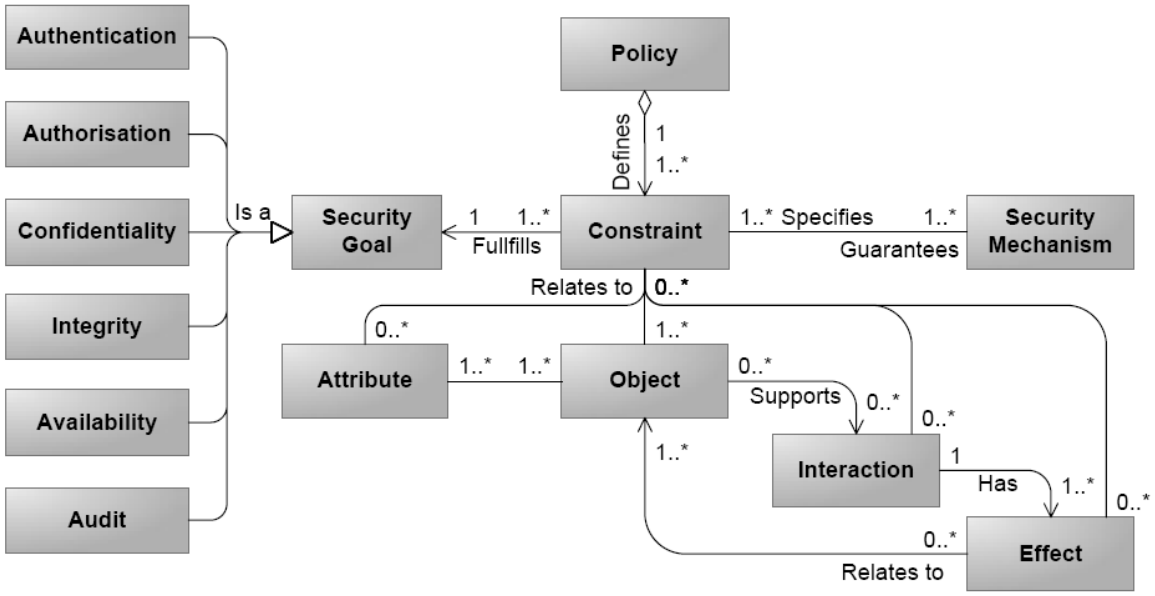


**Figure 4.5: Overview of a semantic description of security goals**

The next step provides the basis for linking security specifications to business process elements. Figure 4.9 gives an overview about the elements of the BPMN model as XSD diagram. BPMN modelling tools like the Intalio|Designer use this structure for representing the process model. Based on this representation which is used in the BPMN modelling tool and the process design framework of Intalio, we prepared tool-support for handling these security annotations during business process design.

We are supporting the following types of security annotations derived from the security goals and their application to business processes: authentication, authorisation, confidentiality, integrity, and audit. The security goal ”availability” will be supported only indirectly.

With this approach and appropriate tool support, we are able to model security properties at the BPMN level during business process design and receive a list of security rules in relationship with the business process elements concerned. Transforming these annotated security descriptions to the security enforcement framework of the project has been implemented. For details about its implementation see Deliverable D3.2 [2].

Our language dealing with these requirements is embedded in the graphical process-modelling language BPMN as structured text annotations. The aim is to represent security constraints in BPs declaratively. Because the BPMN standard provides text annotations, so-called artefacts, these security extensions are standard conform. In this section, we give an overview of the language and describe some selected constraints in detail. The constraints chosen are typical in the BP context. The complete security annotations together with their meaning are described in the Annex 6.

We group the annotation types of our language by security aspects, which we identified in the requirements analysis. These are authorization, authentication, auditing, confidentiality, integrity, and privacy- and trust-related user involvements. The first groups are well-known in literature, but not sufficiently specified in existing approaches for BPs [39], [40], [41], [42], [43]. Annotations for privacy- and trust-related user involvements are a new notion.

We provide security annotations for the following BPMN elements[1]: activities, groups of activities, pools and lanes, data, events, and message flows. Additionally, we introduce the concept of roles. Roles can be assigned to the BPMN elements pools and lanes or to activities (or group of activities). According to the BPMN standard, an activity can be a simple task or a sub-process.

Annotations have the following generic structure:

$$\ll \textit{Annotation term: list(parameter-name="value")} \gg$$

Parameters can be optional. Each annotation term is defined for a particular set of BPMN 2.0 elements, i.e., for activities, lanes, data objects, data stores, or message flows. According to the BPMN standard, an activity can be a simple task or a sub-process.

### 4.2.3 Authorization Constraints

The authorization constraints (Table 4.2) specify who possesses which rights under which restrictions. Dots indicate parameters we have omitted. To manage authorizations, one typically specifies the roles allowed to perform or to access annotated elements.

| Authz Constraint | Syntax |
|---|---|
| Role Assignment | ≪Assignment: type="Role" name="$rolename"≫ |
| Mechanism Assignm. | ≪Assignment: type="Mechanisms" ... ≫ |
| User Assignment | ≪Assignment: type="User" name="$username"≫ |
| Separation of Duty | ≪SoD: role="$rn" number="$nr" ... ≫ |
| Binding of Duty | ≪BoD: spec="weak"... ≫ |
| Adaptation | ≪Adaptation: rights = "$rightsname"... ≫ |
| Task-Delegation | ≪T-Delegation: target="$rolename"... ≫ |
| Data-Delegation | ≪D-Delegation: rights = "$rightsname"... ≫ |

**Table 4.2: Overview of Authorization Constraints**

Authorization constraints, except for delegation for data objects, can annotate all kinds of BPMN activities, pools, and lanes and hold for all activities contained therein. **Assignment** annotations allow to bind roles or users to BPMN elements, so that only role holders or specified users are allowed to execute the activities of these elements. Further, $MechanismAssignment$ allows to specify the ressource-allocation mechanism used, to assign individuals to perform the annotated BPMN activities. The annotations for separation of duty (SOD) and binding of duty (BoD) constrain the execution of a group of activities or of multiple instances of an activity, to be performed by different individuals (SoD) or by the same one (BoD). To avoid process blocking because, say, process participants are absent, we allow to ease BoD constraints by explicit transfer of responsibility, i.e., a weakening of the BoD constraint by means of a re-assignment. The optional parameter $spec =$"$weak$" specifies this. To give way to ad-hoc process changes in a controlled way, we allow the specification of rights to adapt process instances.

*Example:* Figure 4.6 shows two role assignments for lane "placement coordinator" and lane "learner", next to other annotations. This means that these lanes represent roles. The BoD annotations in Figure 4.6 mean that a learner cannot transfer activities to other role holders of learner, while a re-assignment of activities by placement coordinators is possible.

**Delegation:** Delegating access rights may apply to BPMN activities, so-called *task-based delegation*, but also to data objects. To differentiate between these two kinds of delegations, we introduce ≪T-Delegation≫ and ≪D-Delegation≫. We now describe data-based delegation. Process participants can delegate rights to access data objects to someone who currently does not have the authorization. In our context, delegation deals with access rights to data which is external or is controlled by the same process; see the data handling in BPMN 2.0 (Section 10.3 in [44]). Delegation varies depending on the category

[1]Currently, the Intalio|Designer, based on the Eclipse BPMN Modeler Framework, does not provide full BPMN 2.0 support. Therefore, we do not differentiate between specialized subtypes of tasks, e.g., User Tasks, Service Tasks, or Script Tasks, which have been introduced by the BPMN 2.0 specification. A follow-up tool of the Eclipse BPMN Modeler, the Eclipse BPMN 2.0 Modeler, has been proposed as an Eclipse project in April 2011, but has not yet been integrated into the Intalio|Designer or (according to our knowledge) any other BPMN modelling tool so far.

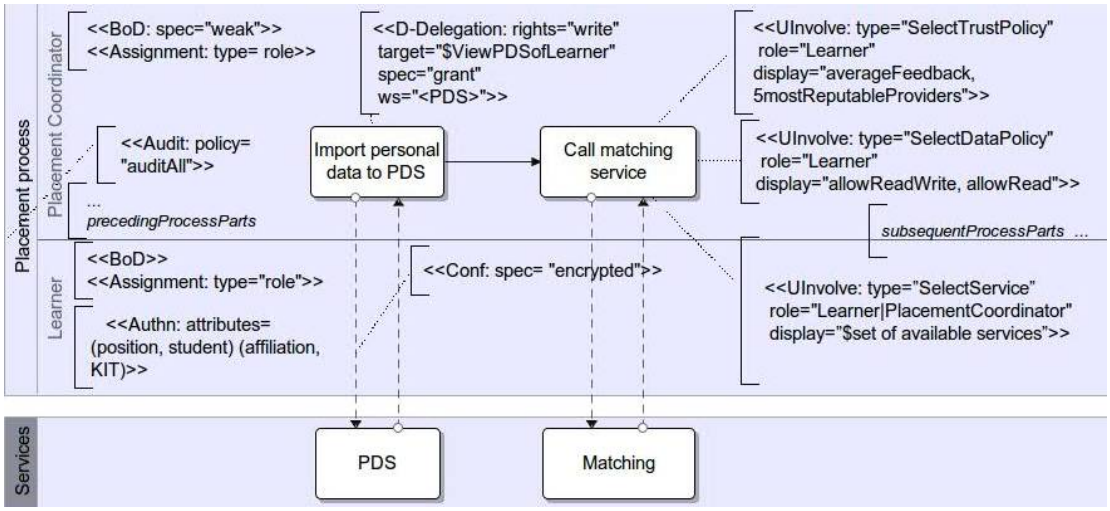


**Figure 4.6: Example Security Annotations**

of the data concerned and therefore requires different parameters. We see two categories: A *data object* is local to the process, and its lifetime is bound to the process where it is defined; *data stores* are external data sources which persist beyond the lifetime of a process. For instance, this could be a database, say, with personal data, like e-health or e-portfolio data. For internal and external data, there can exist access restrictions. Rules to access *data stores* are often specified using external mechanisms. Access rights to *data objects* in turn have to be defined explicitly within the BP. During process execution, role holders who have the right to access the data in question might lose it. Next, external services called need access to external data which a role holder but not the service itself is authorized to. These are situations requiring delegation of access rights.

The annotation

$$\ll \textit{D-Delegation: rights=“\$rightsname” target=“\$objectname”}$$
$$\textit{interval=“(\$activityname1,\$activityname2)|\$group” role=“\$rolename”}$$
$$\textit{poolname=“\$poolname”|“\$lanename” ws=“\$webservicetypename” spec=“\$specification”} \gg$$

specifies the delegation of access rights to data, with $read$, $write$, $delete$, or $policy$ as possible values for rights. If there is a delegation of access rights which are attached to the data object in form of a sticky policy, the $rights$ parameter is optional, but in this case the $target$ parameter is required. The parameters $target$, $role$, $poolname$, $ws$, $spec$ are optional.

The specified rights or the rights of the object specified in the policy are delegated. The rights describe which use of the parameter $\$objectname$ is allowed. The validity of a delegation can be specified for the execution time of activities in the interval [$\$activityname1$, $\$activityname2$], i.e., the part of the process starting with $\$activityname1$ and ending with $\$activityname2$, or in the $\$group$ of activities. The delegate is specified using either an $rolename$ explicitely given or implicitly defined by a $poolname$ or a $lanename$, which are used as rolenames, or a $webservicename$. $spec$ denotes a delegation with $grant$ rights for the delegate.

*Example:* Figure 4.6 shows a process activity dealing with a personal data store (PDS). The annotation ≪ D-Delegation ... ≫ gives access rights to the “$View-PDS-of-Learner” from the holder of role “Placement coordinator” to the web service of type “PDS”.

#### 4.2.3.1 Authentication

The annotation ≪ *Authn: attributes= list (“attributename”, “value”) idp=“\$identity provider”* ≫ means that process participants must be authenticated. The task of an IDP is to certify attributes of process participants. This certification leads to authentications. For this purpose, we allow for specifying a list of attributes which an IDP has to certify. For instance, any holder of role “learner” in Figure 4.6 must be authenticated as a student at KIT and authenticated by an identity provider defined in the annotation.

#### 4.2.3.2 Auditing

The annotation $\ll$ *Audit: policy =*"$\$policyname$" $\gg$
enforces a monitoring of the execution of an activity, a group of activities, a data access, an event, or a message flow. Legal requirements call for different kinds of logging. For example, if a process handles personal data, it must be logged, among others, who has performed which action upon which data, and at which time [45]. Thus, we assume that an auditing policy (parameter $policy$ ="$\$policyname$"), stored in the BPMS, specifies the objects to be logged and the kind of logging. To illustrate, we have annotated the pool "placement process" with the annotation term $Audit$ in Figure 4.6. This specifies a monitoring of all task executions and all security tasks, e.g., the delegation or the encrypted call of the matching service according to the policy "auditAll".

#### 4.2.3.3 Confidentiality and Integrity

Authorization mechanisms support a basic level of confidentiality, because only authorized humans or services may perform activities and access data. To improve confidentiality the system can also protect message flows. This can be expressed by $\ll$*Conf*$\gg$.

$$\ll \textit{Conf: spec="\$value"} \gg$$

We allow the parameter "spec". Its possible values are "encrypted" or "signed". For example, the data flow from activity "Import personal data to PDS" to activity "PDS" in Figure 4.6 must be encrypted. This addresses the shortcoming of existing security vocabularies exemplified in the introduction.

The annotation $\ll$*Integrity*$\gg$ means that data integrity must be enforced. We allow the annotation of data (i.e., data objects, data stores, data inputs, data outputs) and message flows.

#### 4.2.3.4 User Involvements

| User Involvement | Syntax |
|---|---|
| Give Consent | $\ll$UInvolve: type="Consent" ... $\gg$ |
| Service Selection | $\ll$UInvolve: type="SelectService" ... $\gg$ |
| Select Data Access Policy | $\ll$UInvolve: type="SelectDataPolicy" ... $\gg$ |
| Select Trust Policy | $\ll$UInvolve: type="SelectTrustPolicy" ... $\gg$ |
| Give Trust Feedback | $\ll$UInvolve: type="TrustFeedback" ... $\gg$ |

**Table 4.3: Overview of User Involvements**

We have identified so-called user involvements to support user concerns, e.g., privacy concerns of data owners. These user involvements are important, because users have to specify and agree to privacy and trust preferences that are context-specific at runtime. Having examined various application scenarios, we have identified the following user involvements: giving consent, selecting services in line with the trust levels required, specifying data-access policies or trust levels. There is also the need to allow to give feedback about web-service calls, in order to be able to compute the trust level of these services later on. To support the representation of user involvements, we introduce annotation terms to specify them declaratively. One can annotate activities and certain events of a process model with user involvements. Thus, the process designer simply specifies them instead of modelling them explicitly. The representation is as follows:

$$\ll \textit{UInvolve:list(parameter-name=value)} \gg$$

Each user involvement has a parameter of name *type*, and its value specifies the intention of a user, e.g., *type="consent"* means giving user consent. Unless specified otherwise, these involvements are invoked right before the annotated activity takes place. Otherwise, there is a parameter $insertplace$, indicating the position of the involvement in the sequence of activities with the activity the fragment has to be inserted

before. Our constraints (Table 4.3) allow to specify when during process execution users can set their privacy preferences.

**Select Trust Policy.** It is intuitive to characterize service providers by trust levels, e.g., their reputation gained in the past. Process participants can specify the minimum trust level of providers by means of a trust policy. The following annotation specifies the selection of a trust policy:

$$\ll \textit{UInvolve: type="SelectTrustPolicy" display="list(option)"} \gg$$

*Example:* The user involvement of type "SelectTrustPolicy" in Figure 4.6 means that a learner has to select a trust policy from the options given.

## 4.2.4 Transformation of Security Annotations

In this section, we describe how we transform our security annotations of process models to the execution level. A BPMS enhanced with BP-specific security components, see Section 4.2.4.1, executes the BPs. Depending on the nature of the security constraints, we have identified three kinds of transformation targets. We explain the specifics in Section 4.2.4.2.

### 4.2.4.1 Secure Business-Process-Management System

A common security framework, but without explicit BP support, is XACML [46]. It is a declarative access-control policy language. The processing model of XACML contains the following components: a policy-decision point (PDP), to check against security policies whether certain operations are allowed, a policy-enforcement point (PEP) to intercept access requests and to enforce the decision of a PDP, and a policy-information point (PIP) to provide further security information for PDP and PEP. In addition to the XACML framework, a security framework in a heterogeneous environment requires further functionality. There, authentication typically relies on federated identity management with IDPs. Finally, a trust PDP to manage trust and an auditing bus to support auditing are further components of the security framework we rely on.

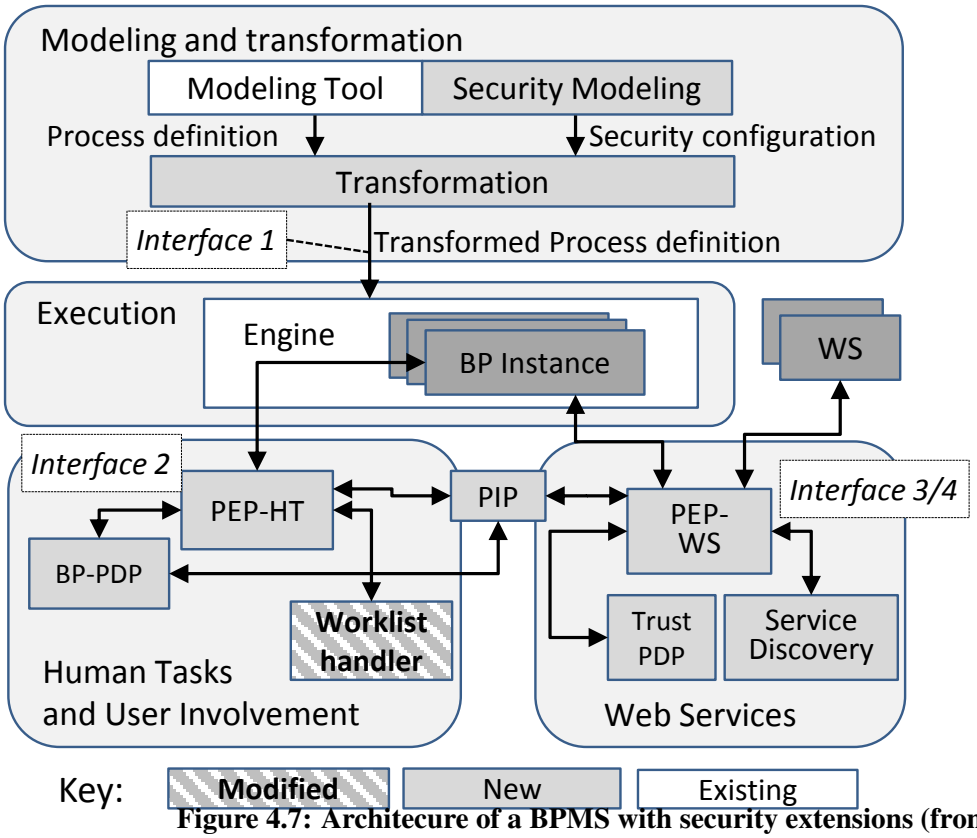


**Figure 4.7: Architecure of a BPMS with security extensions (from [1])**

The security requirements of Section 3.2 necessitate the refinement of the components of the XACML processing model by BP-security-specific aspects. We refer to the new BP-specific components as PEP-

WS, PEP-HT, BP-PDP and PIP. In particular, the components have to take into account the history and context of the process instance, to, say, bind duties to a specific role holder, or to ensure that access rights hold exactly for the execution time of the activities in question. Further, delegating access rights is essential for service calls handling private data which the process performs for a user. In this case, the BP has to transfer the necessary rights to the service called, e.g., to access an external data store.

The architecture in Figure 4.7 (see [1] for details) contains the main components and interfaces of a BPMS according to the WfMC reference model [47]: the BP *Engine* in the middle connected with interfaces to the other components, the *Web Services* component dealing with calls to web services and to other processes (Interfaces 3 and 4), and the *Human Tasks* component (Interface 2) with a worklist handler and binding of client applications, i.e., typically web interfaces to users and a user interface with a user-specific worklist. The *Administration and Monitoring Tool Component* is not part of this figure. Interface 1 connects the modelling component to the engine.

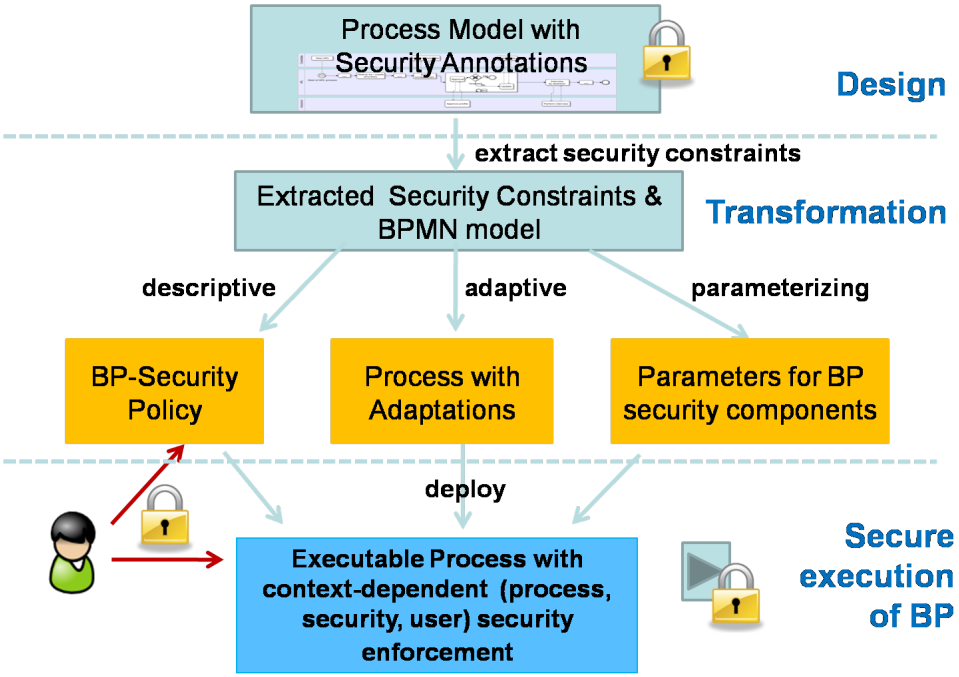


**Figure 4.8: Steps of the Life-Cycle of Secure BPs**

To give way to security-specific functionality, in our team we have extended a WfMC-compliant BPMS architecture as follows [1]: A dedicated security component, the BP-specific PEP, handle all interactions with users (PEP-HT) and with external web services (PEP-WS), as a proxy of the engine. Before any web-service call and any execution of human tasks, the PEP-WS checks if access rights are sufficient, if the transport has to be encrypted etc. The same holds for incoming service calls.

Security modelling complements the BP modelling. The resulting security-enhanced BP model is input for our transformation component which we present next. Figure 4.8 gives an overview of the data involved in this life-cycle of secure BPs.

The *design phase* results in a BPMN model with security annotations. To deal with security annotations, we have developed a dedicated tool, the Security Modelling and Configuration Component for BP (BP-SMC). In the *execution phase*, the secure BPMS runs instances of the resulting process enforcing the security constraints. In the following, we describe the *transformation phase*. We are using the Ecore BPMN model to interpret the process specification of the graphic modelling tool, and filter out the security annotations.

#### 4.2.4.2 BP-Security-Model Transformation

The main concern of this phase is the transformation of an annotated BP model to representations which support the enforcement of the annotated constraints during process execution. To this end, we have to deal with several system layers described by distinct models. Model transformation is essential for any model-driven software development (MDD) [48]. MDD approaches for security in BPs, see [49, 50, 51, 52]

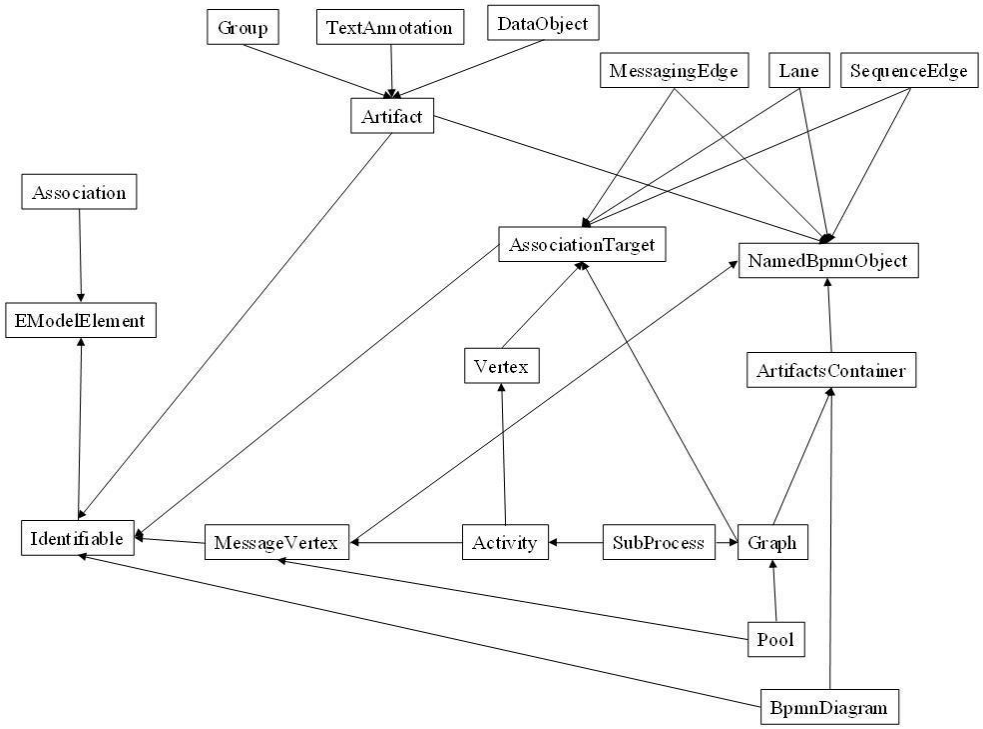


**Figure 4.9: Hierarchical structure of BPMN process description.**

in particular, have, e.g., generic process models and security models as source, and XACML policies or UML use case models as targets of the transformation. Our transformation approach starts from a BPMN process model and a security model, i.e., the annotations. The BPMN meta model, i.e. the source model of our transformation, is expressed as a Meta-Object Facility (MOF) model [53]. We have discovered that different security constraints on BPs need to be transformed in different ways, i.e., have different outcomes depending on the security category of the annotation. A transformation has at least one of the following effects: generating or modifying a BP-access-control security policy, setting parameters to configure security components, or adapting the BP by canned process fragments. This leads to the three target models: XACML as policy language enhanced with BP-specifics, a data model of configuration parameters for the security components of our secure BPMS, and a BPMN process model adapting the model of the application process.

#### BP-security policy

A security policy consists of rules, such as authorization rules or rules protecting the data flow, e.g., with delegations. BPs control the sequence of tasks. Thus, general security-policy languages such as XACML are not sufficient. The enhancements include security rules for tasks and flows of tasks, and constraints related to the process status. We refer to such an enhanced policy as BP-access-control security policy or BP-security policy, in short. We argue that one should map these security annotations to policies the extended PDP can handle.

The transformation in this category results in an XACML policy, enhanced with BP-specific constructs: In particular, security constraints may refer to tasks and relationships between tasks, as well as to the execution history, like BOD, and the execution status of the BP. During process execution, the BP-PDP checks any access to tasks for conformance to this policy.

*Example:* Think of a BoD constraint that defines the part of the process BoD is valid for and the role it holds for. The transformation of annotation ≪BoD: spec="weak"≫ of the Placement Coordinator lane in Figure 4.6 generates additional rules that become part of the BP-access-control security policy:
<*TaskGroupSettings*>

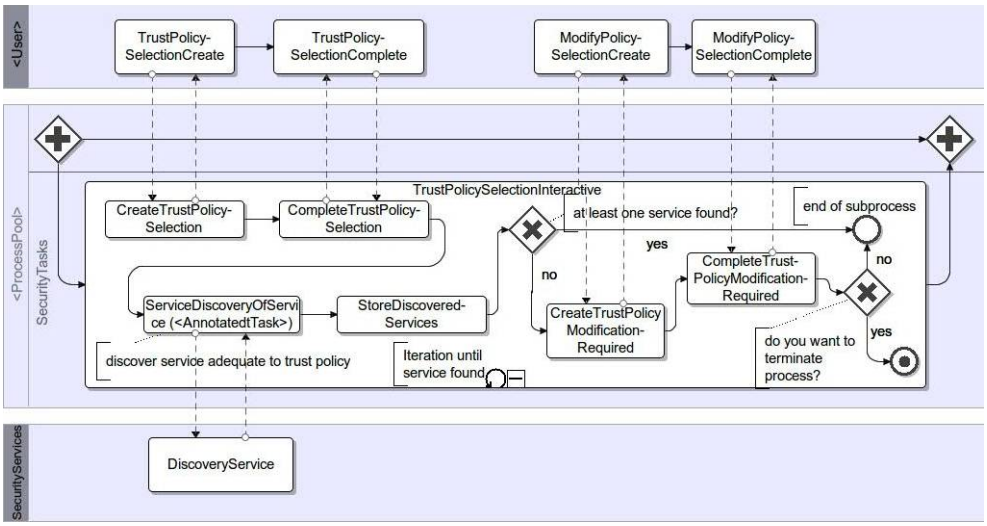


**Figure 4.10: Process fragment implementing the trust policy selection and the service discovery loop**

*<TaskGroupSpec TargetID="PlacementCoordinatorLane"*
*BoD="true" weakBoD="true" />*
*</TaskGroupSettings>*

When assigning a holder of role "placement coordinator" to a human task, the PEP-HT in interaction with the worklist handler calls the BP-PDP to check if the assignment envisioned observes the access rules of the BP-security policy, e.g., in case a BoD constraint is affected.

#### Configuring security components

The second kind of transformation concerns the parameterization of calls to security-specific components other than the BP-PDP. These components, PEP and PIP in particular, take process context like validity periods of security rules, history of the process flow or role holders involved in the process into account. The PEP comes into play, because calls going to or coming from external components or human tasks are tasks the secure BPMS first calls the PEP for. The PEP then executes the security checks required, e.g., a PDP call to check access rights to parameters of the call. Further, it configures the call with security parameters, e.g., a WS-Security header with encryption parameters. The BP-SMC passes information taken from the annotations to the PIP, to facilitate parameterization of security calls issued by the PEP. If a component is responsible for compliance with an annotation, transforming the annotation yields the parameters of the invocation of that component.

*Example:* ≪Authn: ...≫ will result in an authentication call to an IDP. The annotation and its context information yield the identifying attributes of the role holder in question. This information is stored in the PIP so that it can be used in other PEP calls during process execution. As another example, think of a ≪Conf≫ annotation of a message flow to a web-service task. It specifies parameters to invoke the web service in an encrypted manner or with a digital signature, see Figure 4.6.

We have identified a set of security-configuration parameters required to call the security components fo the secure BPMS. The design of our security-annotation language from Section 4.2.2 has taken these configuration opportunities into account. Examples of the security components are an IDP, service discovery using security and trust constraints, an audit bus for logging, a trust PDP, or a security PDP of an external data store. In consequence, the PEP can set WS-Security-compliant parameters for the WS call as well as parameters for calls to other components.

#### Process adaptation

The third kind of annotation needs an explicit realization with several activities to enforce it. It must be tailored to the sequence of the BP, and it can involve process participants. In addition, calls to security components, e.g., PDP checks, calls to service discovery etc., can be necessary with this kind of annotations. In particular, this is the case for ≪UInvolve≫ annotations. Another example is the ≪Audit≫ annotation. It results in a BP task that calls the PEP to initiate a call to the audit bus to log the execution

of the annotated BPMN element. Note that the logging call is parameterized, and the parameter values are an outcome of the transformation of the respective annotation. The parameters have been stored in the PIP.

We deal with this type of annotation by adapting the process model, i.e., by plugging process fragments into it, with security-specific activities. For each ≪UInvolve≫ annotation, there usually exists more than one fragment, depending on the behaviour required. I.e., there is the need for semantic specification in order to select an appropriate one.

*Example:* In the process model from Figure 4.6, a "Learner" might allow web services to match her personal e-portfolio data with job offers only from trusted web services. To take trust into account for the service selection, the process designer can now annotate the activity with a ≪UInvolve≫ constraint of type "SelectTrustPolicy". It specifies that the "Learner" can select one of the options listed as display parameter:

≪ *UInvolve: type="SelectTrustPolicy" display="trust All, average Feedback, most reputable Users, 5 most reputable Users"* ≫

To illustrate, Figure 4.10 shows the process fragment which is inserted into the BP before the annotated activity "Call Matching Service". It contains the following activities and user interactions:

- A user interaction, i.e., tasks to prompt the user. The web form, which is provided together with the process fragment, must contain (links to) the trust policies available to be checked during the transformation. They have to be parameters of the annotation. This requires a consistency check, which is part of the process fragment (but, due to space problems not contained).
- The DiscoveryService task causes a PEP-WS call of a web-service discovery with the trust policy selected as parameter.
- If there is no service with that trust level, there is a further iteration of trust-level selection and service discovery.

Using pre-defined BP fragments brings several benefits: The process designers do not need to be familiar with the security components. Next, providing such fragments supports the correct implementation of the annotations.

## 4.2.5 Knowledge Annotator for Modelling Secure Business Process Models

The knowledge annotator tool has been designed with the purpose to assist the business process modeller into modelling secure business process models. More precisely, the annotator ensures the correct specification of the security annotations that are built on top of classical business processes. It uses a (lower common) ontology representing the security constraints for business processes and a knowledge base storing a set of previously defined annotations. The knowledge base captures and stores the user knowledge for further retrieval and reuse (e.g. recommendations, statistics, etc.).

The current work is being derived, adapted and continued from the research outside TAS$^3$, hence it constitutes background for the project. We describe here the conceptual design of the annotator, which will be technically specified in Deliverable D3.2 [54].

### 4.2.5.1 Approach

The knowledge annotator is designed to be user-friendly, acting as an intelligent system. It captures the user's modelling intentions and presents him with recommendations, after a preliminary analysis (matching) performed on the predefined security constraints retrieved from the knowledge base. The gap between the user's domain and the security domain is thus bridged, through an ontology of security constraints, as illustrated in Figure 4.11.

The security annotations are classified into several security categories: Auditing, Authenticity, Authorization, Data and Messageflow Security, Delegation and User Interaction (as shown in Figure 4.12).

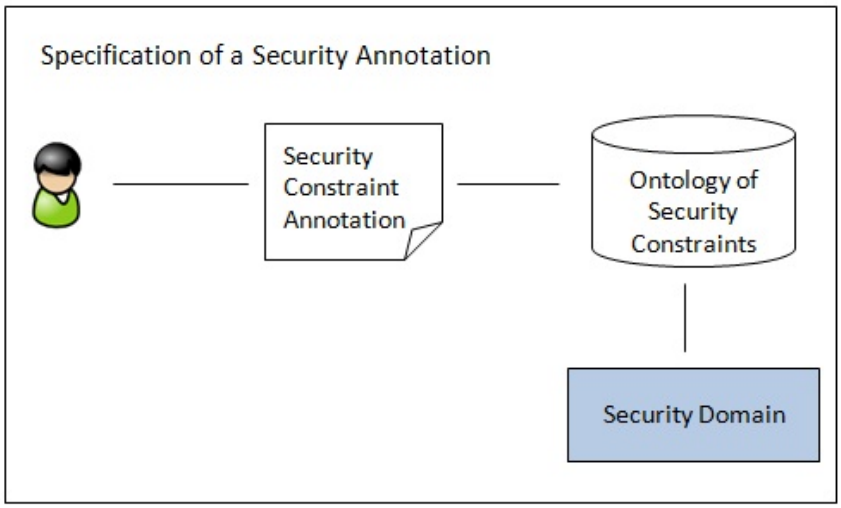


**Figure 4.11: Bridging the Gap between the User and the Security Domain**

Figure 4.13 illustrates the concept of security annotation. A security annotation is specified by an Annotation term, followed by a list of parameters with their corresponding values:

*≪Annotationterm:list(parameter="value")≫*.

A security annotation can refer to more than one BPMN element.

The basic data element used by the knowledge annotator is represented by the Security Constraint, Element, Parameter, Value (SCEPV) object. The SCEPV object encapsulates the four entities needed to completely define a security annotation: the security constraint, the BPMN element being annotated, the parameters and their corresponding values.

For every security annotation that the user intends to define, he must indicate as many hints (SCEPV elements) as possible (according to his knowledge) corresponding to what he has in mind. The user input is captured by the system and analyzed in order to make recommendations to the user. After performing the analysis, the system returns to the user the SCEPV elements that are related to his initial (usually incomplete) specification.

After a SCEPV object is completely defined by the user, it is stored by the system in the knowledge base and will be used for knowledge retrieval when the user asks the system for recommendations. Note that the first three entities in every possible SCEPV object (i.e. security constraint, BPMN element and parameters) are already stored in the ontology base, together with the relations they share with respect to one another (see the ontology for security annotations, specified in Deliverable D2.3 [55]).

When values are associated to parameters for a specific security annotation defined by the user, we say that the security annotation ontology for that particular security ontology is instantiated. It is at this point that the knowledge base is capturing and storing the user knowledge (the way the user chooses to instantiate the security annotation).

We envisage the ontology annotator as a tool assisting the user with a six-step process, supported by six components of the annotator (see Figure 4.14):

1. Capturer
2. Annotator
3. Indexer
4. KB Retriever
5. Comparator

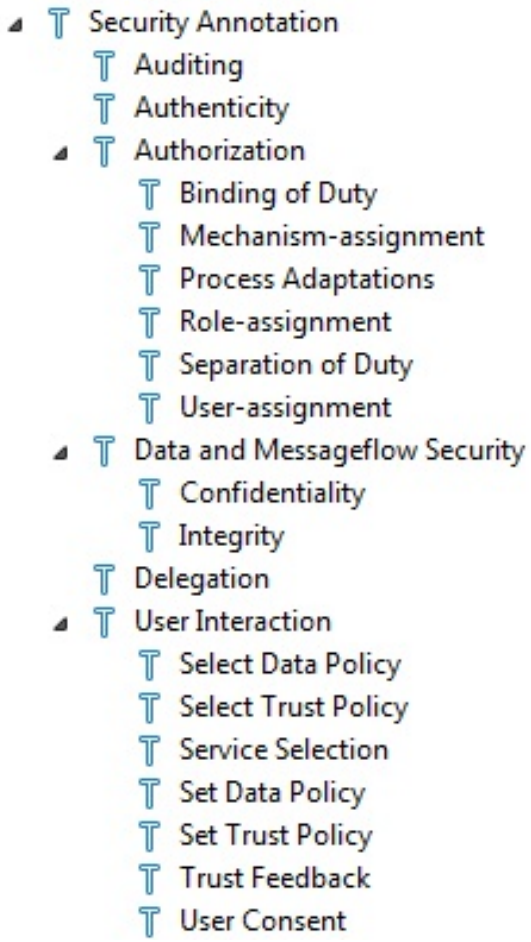


**Figure 4.12: Taxonomy of Security Annotations**

6. Presenter

The functionality of the six components is described as below:

#### Capturer

This component captures the user design intent and transforms it into an SCEPV object. The SCEPV objects are passed from one component to the other along the process. The user input is captured thanks to a friendly user interface. The UI presents a template with several fields for the user to textually specify his rules. The rules represent the abstraction of the security constraints. At this stage, the user can be assisted by an ontology browser.

#### Annotator

The SCEPV object is annotated in a semi-automatic manner with concepts from the ontology base (OB).

#### Indexer

The Indexer is used for the KB storage and retrieval of SCEPV objects.

#### KB Retriever

This component retrieves similar fragments (SCEPV objects) from the knowledge base. The knowledge base contains semantically annotated patterns (SCEPV objects) for the rules (i.e. the security constraints).

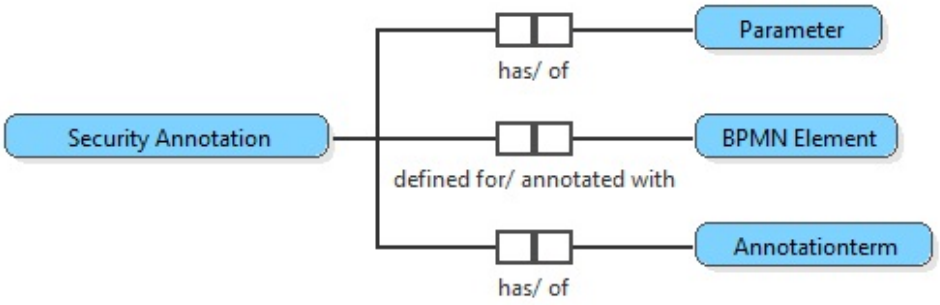


**Figure 4.13: Representation of the Security Annotation concept**

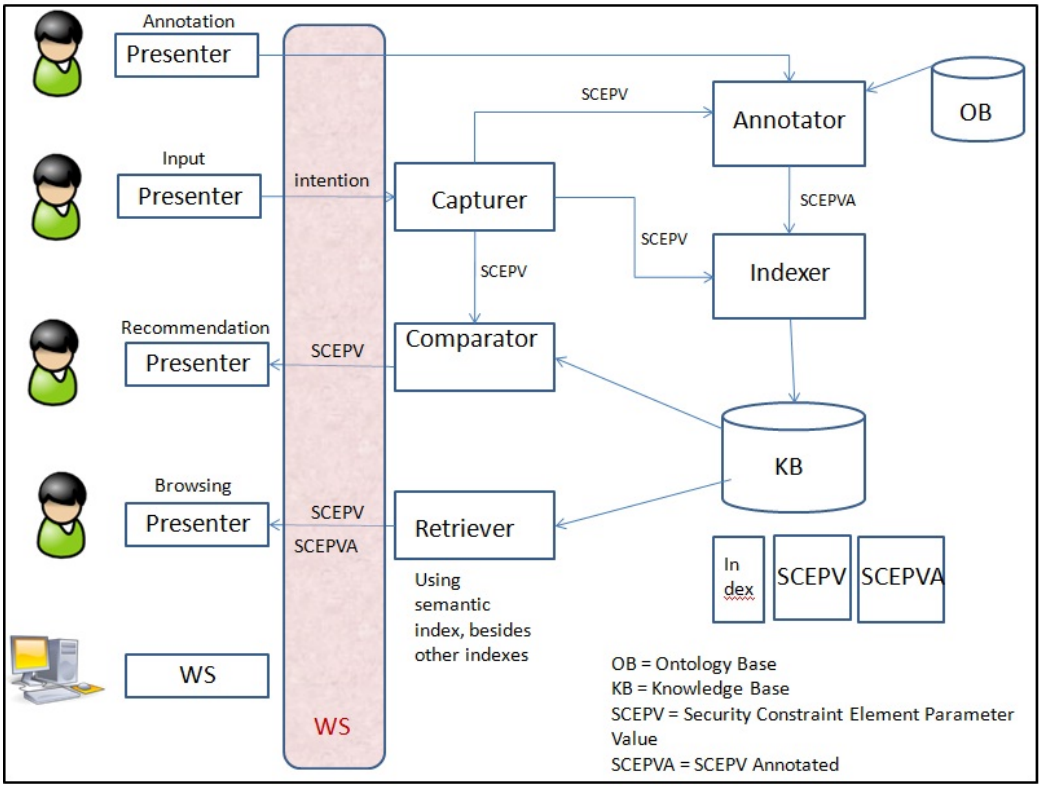


**Figure 4.14: Knowledge Annotator Architecture**

#### Comparator

The Comparator component compares the user demand (SCEPV input object) with the design choices retrieved from the knowledge base in the previous step.

#### Presenter

The role of the Presenter is to present the user with the design choices (SCEPV objects) retrieved from the KB.

### 4.2.5.2 Use Cases

Based on these functions, there are several possible use cases for the knowledge annotator (also depicted in Figure 4.14):

#### Use Case 1

The first use case represents the basic use case for the knowledge annotator, when the user is interrogating the ontology base to retrieve and browse the correct concepts he needs in order to define a specific security annotation. The user can ask the system to make faceted search on the BPMN elements, on the security annotation types and on the parameters. For example, the user creates a security annotation of the type Authorization which applies to a BPMN element of the type Pools and Lanes. In order to do so, he asks the system to browse all the existing security annotations which are satisfying this condition. The system is performing a query and returns the following results to the user:

- Security Annotation types
  - Role-assignment
  - Mechanism-assignment
  - User-Assignment
  - Separation of Duty
  - Binding of Duty
  - Process Adaptations
- BPMN elements: Pools and Lanes, defined by the user
- Parameters
  - Type
  - Name
  - Role
  - Number
  - Threshold
  - Spec
  - Rights
  - Pre
  - Post

Following these options, the user either decides to make a choice or launches new queries in case the results do not correspond to his modelling intentions. If, for example, the user decides to annotate Pools and Lanes with the Mechanism-assignment security annotation, the result will look like:

- Security annotation: Mechanism-assignment
- BPMN element: Pools and Lanes
- Parameters: Type, Name

#### Use Case 2

A second use case represents the situation when the user needs to instantiate the security annotation. In this situation, the user asks the system to interrogate the knowledge base for similar annotations.

#### Use Case 3

The third use case is the situation when the system is performing matching operations between the user input and the knowledge base content in order to retrieve similar instantiated security annotations and statistics about them (e.g. the most frequently used BPMN element, etc.).

#### Use Case 4

The fourth use case is represented by the situation when the user has created the security annotation by memory and needs to check for its syntactical correctness. In this case, the system performs similarity measures between the input and the ontology base and presents the user recommendations.

#### Use Case 5

The fifth use case represents the situation when the user asks the system to check an instantiated security annotation for correctness. In this case the matching is done at the knowledge base level.

#### Use Case 6

The sixth use case represents the case when the actors are two web services which are interoperating via an interface. This situation represents the case of fully automated annotations. The user is not involved in the annotation, but results could be submitted to the human expertise at a later stage.

Use case number one is actually in progress. The remaining use cases are planned for future work on evaluating this approach.

## 4.3 High-Level Ontology Covering Business Processes

In TAS$^3$, ontologies should allow communication across web services and business processes based on semantics rather than syntax. As a result, we would be able to generate business processes based on organisational web services and transform them into BPM graphical representation language for the end user. Moreover, we would be able to discover, substitute, compose and execute web services automatically on the web. Finally, the different components of the TAS$^3$ architecture should be annotated with security and privacy concepts.

As part of this work, we have developed ontologies based on the DOGMA (Developing Ontology-Grounded Methods and Applications) ontology framework [56]. According to the double articulation philosophy, a DOGMA ontology consists of a lexon base (i.e. intuitive conceptualisation), and a layer of reified ontological commitments. Its double articulation principle and grounding in natural language representation of knowledge makes DOGMA particularly fit for representing business-level as well as technical terminology and semantics typically found in business process models and their web-service implementations respectively.

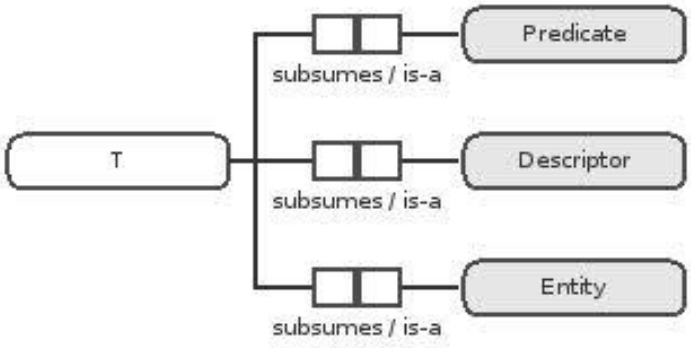


**Figure 4.15: The top layer of the DOGMA Upper Ontology**

In Deliverable D2.2 [57], we have described the DOGMA upper ontology. In brief, the top layer of the upper ontology consists of three main concepts (Figure 4.15), namely Predicate, Entity and Descriptor. A predicate denotes a verb which affirms or denies information about the subject. For example, a task in a business process is represented as a type of predicate. An entity represents anything that can take part in

an action or that can be acted upon. For example, PIIs are a type of entity. Finally, a descriptor categorises or describes either an entity or a predicate. For example, we would use a descriptor to represent sensitive information.

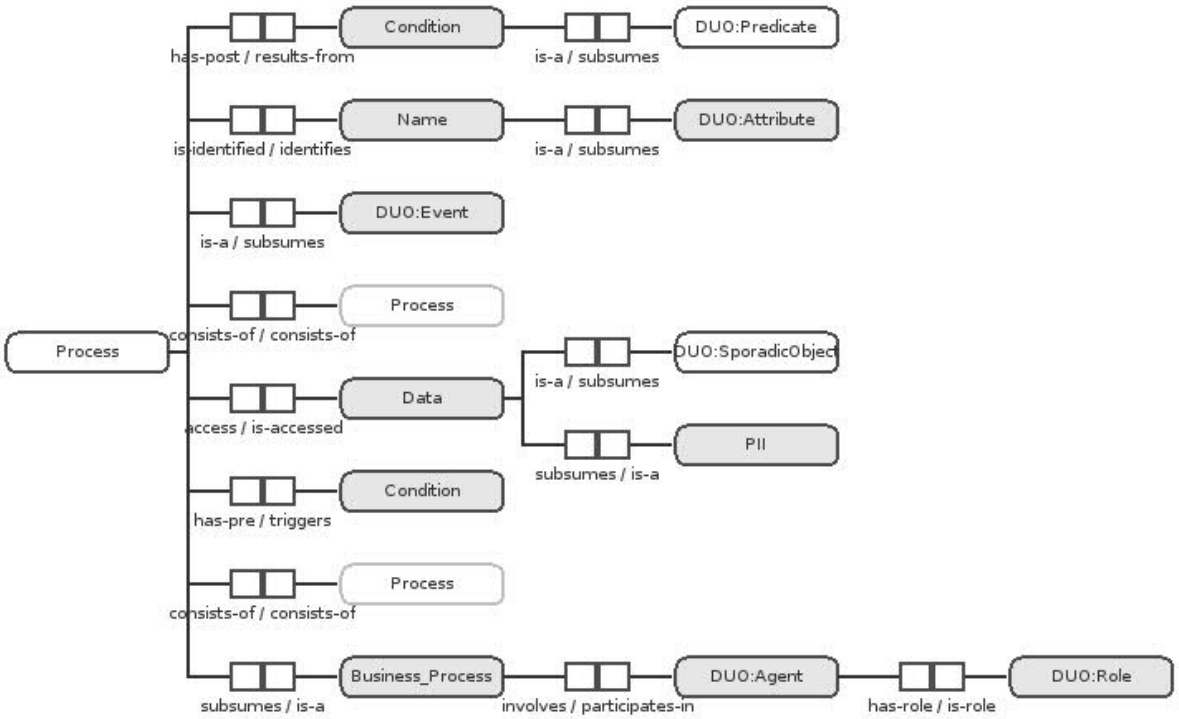


**Figure 4.16: Representation of the business process concept in lexons**

Figure 4.16 represents the Process concept in lexons from which the properties of a Business Process are inherited. For instance, a process consists of one (or more) processes, and each process is triggered by a condition (i.e. pre-condition) and results in other conditions (post-conditions) that may or may not be the trigger of other processes. For example in Figure 3.2, the event to ”Receive Contract” triggers the ”Input Candidate Data” process, which itself results in ”Finalising Candidate Data”. Note that the pre-conditions can also include authentication and authorisation policies.

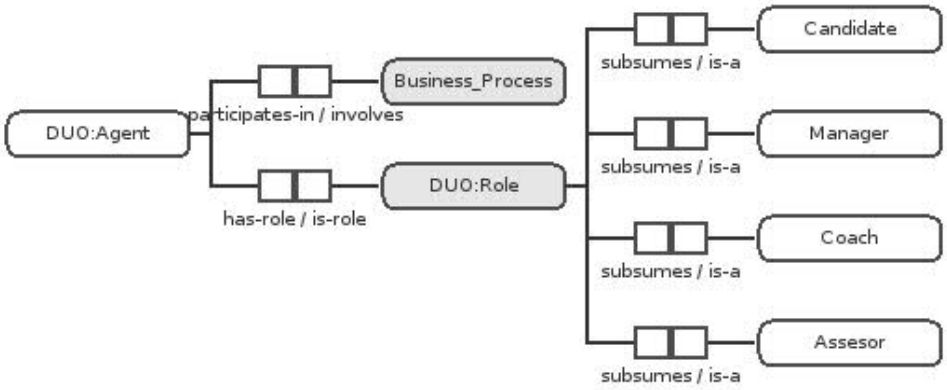


**Figure 4.17: Roles in the Kenteq APL process**

Using the DOGMA approach, we can extend the different concepts to include application specific data. For example, we can define the different types of roles available within the APL process (Figure 4.17). Based on these concepts, we can declare that an Assessor assesses the progress of an assigned candidate.

Similarly, we can define the concepts related to PII (Figure 4.18). This will enable web services to know whether they have to consider a security aspect in respect to the data. The main purpose of this

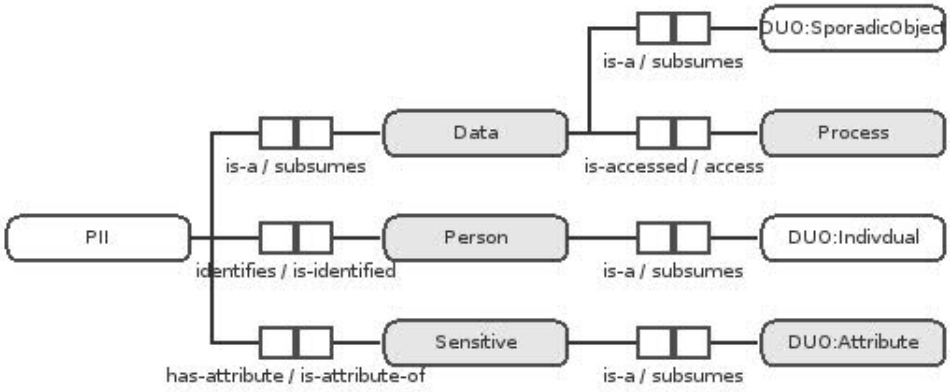


**Figure 4.18: Representation of the Personally Identifiable Information concept**

representation is to define the things that authorisation policies are going to be applied to. For example, a business process accessing a patient's medical records would have to define the rights that a user needs to fulfil to access it. In TAS$^3$, the dynamic adaptation of business processes will use information related to security, privacy and trust (see [57] for more details on the security and privacy ontologies).

## 4.4 Security Policies for Business Processes

Business processes allow expressing several perspectives and their relationship. Several aspects have been described in the literature [58, 59]: The *functional* aspect describes the structural composition of processes with functional components, i. e., tasks and sub-workflows. The *informational* aspect describes the data flow in the process. The *behavioural* aspect deals with the control flow with causal and temporal relationships. The *operational* aspect regulates the involvement of tools and applications in a workflow. Finally, the *organizational* aspect describes the relationship to organizational structures and who performs the operations. Security has also been mentioned as an aspect, which is particularly important in TAS$^3$. We will describe security concepts combining several perspectives on a business process in the following sections.

In the modelling phase, security specifications are expressed as annotations to the BPMN model of the process, see Section 4.2. The BPMN model is translated into an executable business process in the WS-BPEL format which is then executed. The security annotations have to be translated as well. Some annotations are translated into integral parts of the business process, see Section 4.6.3, while others are translated into a process policy. A third target is to store configuration parameters for calls of security components. For a further description see also [60].

Note that security annotations of a business process are not the only source for the content of business-process policies. Furthermore, trust policies must be integrated into the business process policy to enable an appropriate selection of Web Services. In addition, the relationship between business-process policies and other policies needs to be considered. Users' specified data access policies for the TAS$^3$ trust network must be adapted accordingly to a business process policy. Another example are data policies users can specify particularly for a business process by means of a user request (see Sec. 4.2.2). Furthermore, trust policies must be integrated into the business process policy to enable an appropriate selection of Web Services. This can lead to several problems. For example, there might be contradictions in data policies, specifications on different granularity, and even missing information.

Business-process policies comprise the following settings:

- Prerequisites a user must fulfill to be eligible for performing a task or a group of tasks, i.e. roles or attributes that he/she must possess (cf. Section 4.7.2).
- The assignment strategy specifying how a user is chosen if more than one is eligible. Our concepts are flexible to integrate different assignment strategies. The realization of such strategies is not the focus of our work. The main reason for this is that the assignment is not security-related, and thus

outside the main focus of TAS$^3$.

- Role hierarchy specifications.
- Binding of Duty (BoD) and Separation of Duty (SoD) constraints (Section 4.7.5).
- Rules for selecting the identity to use in a web-service call.
- Rules for service selection (e.g. trust policy).
- Specification of data access rights (e.g., for PII data).
- Process elements that need to be monitored during process execution.

Our preliminary policy format is PERMIS-based. The aim is to re-use parts of existing policy languages where possible. The PERMIS policy language is used in TAS$^3$, so we analysed which parts of a PERMIS policy can be re-used in business process policies:

- The *Subject Policy* specifies the domains users may belong to so that they are eligible for access. This is a quite technical setting relating to the positition of the service provider in the trust network, so we think it is inappropriate to set in every business process. We rather see this as a configuration option that is set per service provider.
- The *SOA* (Source of Authority) *Policy* specifies trusted IdPs. This is appropriate for a SP-wide configuration option, as well.
- The *Role Hierarchy Policy* specifies supported attributes and their relationship. This can be extracted from the annotated process model and should thus be part of the business process policy. If no dominance relationship is specified in the process model, the hierarchy will be flat. Note that despite the name, this part of the policy specifies a hierarchy between different attribute values of an attribute.
- The *Role Assignment Policy* specifies which IdP may assign which role types. Again, we think is should be subject to a service-provider-wide configuration.
- The *Target Policy* lists and names possible targets. In a business process model, the targets are groups of tasks for which access control settings have been made. However, as there is no clear syntax for specifying groups of elements as targets, we will use our own syntax here.
- The *Action Policy* specifies which action types are supported by the policy. We currently only support one action, “execute”. This is specific to our implementation, and cannot be overridden at the SP level.
- Finally, the *Target Access Policy* specifies which roles (or attributes) a user must possess in order to be allowed to perform an action on a target. This is directly derived from the assignment rules in the process diagram.

In addition to parts taken from PERMIS, our policies comprise the following:

- Groups of tasks (used as targets) are given a name, and consist of a list of uniquely named tasks. It does not matter which constructs are used to specify these lists on the modelling level.
- Binding of Duty is an attribute of a group of tasks, while separation of duty can be specified between two groups of tasks.
- The identity to be used in a web-service call is selected by including a task calling a web-service in a group with human tasks and specifying binding of duty.

- For now, we provide a string to be used as the trust policy in web-service call tasks. It is not yet specified how to pass trust policies to the Trust PDP during service selection. Additionally, we are planning to automatically involve the user into service selection, so further changes will become necessary here.
- The strategy for assigning users is also an attribute of a group of tasks. We are investigating in the support of different assignment strategies: When there is more than one possible user for a given task, the specified assignment strategy is chosen. This can be a simple random assignment, a strategy such as First-In, First-Out (FIFO), Earliest Due Date (EDD), among others [61]. Further, application-specific mechanisms can be assigned, as described in Section 4.2.2.

A policy looks like:

```
<BusinessProcessPolicy>
  <!-- RoleHierarchyPolicy has the same format as in PERMIS. -->
  <RoleHierarchyPolicy>
    <RoleSpec OID="1.3.6.1.4.1.5923.1.1.1.7"  Type="eduPersonEntitlement">
      <SupRole Value="Administrator">
        <SubRole Value="Staff"/>
      </SupRole>
      <SupRole Value="Staff">
        <SubRole Value="Student"/>
      </SupRole>
      <SupRole Value="Student"/>
    </RoleSpec>
    <RoleSpec OID="1.3.6.1.4.1.5923.1.1.1.1"  Type="eduPersonAffiliation">
      <SupRole Value ="Nottingham" />
      <SupRole Value ="Kent" />
      <SupRole Value ="CV_Service"/>
      <!-- Add an affliation for each entity that should be  able to access
       the data -->
    </RoleSpec>
    <RoleSpec OID="99999.0.1" Type="courseName">
      <SupRole Value ="History"/>
      <SupRole Value ="English"/>
      <SupRole Value ="Maths"/>
    </RoleSpec>
    <RoleSpec OID="1.2.826.0.1.3344810.1.1.14"  Type="courseName">
      <SupRole Value="Co831"/>
    </RoleSpec>
  </RoleHierarchyPolicy>
  <!-- Similar to PERMIS TargetPolicies -->
  <TargetPolicy>
    <TargetDomainSpec ID="JobSeekerTasks">
      <Include TaskId="InitiateProcess"/>
      <Include TaskId="ChooseJob"/>
      <Include TaskId="RegisterPlacement"/><!-- a web-service call -->
    </TargetDomainSpec>
    <TargetDomainSpec ID="PlacementProviderTasks">
      <Include TaskId="ConfirmPlacement"/>
    </TargetDomainSpec>
    <!-- a usual policy will contain a lot more tasks -->
  </TargetPolicy>
  <TaskGroupSettings>
```

```
    <TaskGroupSpec TargetID="JobSeekerTasks" AssigmentMechanism="default"
     BoD="true" weakBoD="true" />
  </TaskGroupSettings>
  <SoDPolicy>
    <SoDSpec>
       <!-- Specifies SoD between the tasks belonging to the two groups. -->
      <SoDPart TargetID="JobSeekerTasks" />
      <SoDPart TargetID="PlacementProviderTasks" />
    </SoDSpec>
  </SoDPolicy>
  <ServiceSelectionPolicy>
    <!-- Defines the trust policy to be used in selecting a service for the
    RegisterPlacement web-service-call activity. -->
    <ServiceSpec TaskId="RegisterPlacement" SelectionPolicy="dummypolicy" />
  </ServiceSelectionPolicy>
  <TargetAccessPolicy>
    <TargetAccess ID="ExecuteJobSeekerTasks">
      <RoleList>
        <Role Type="eduPersonEntitlement" Value="student" />
      </RoleList>
      <AllowedAction ID="EXECUTE"/>
    </TargetAccess>
    <TargetAccess  ID="ExecutePlacementProviderTasks">
      <RoleList>
         <Role Type="eduPersonEntitlement" Value="staff" />
       </RoleList>
       <AllowedAction ID="EXECUTE"/>
    </TargetAccess>
  </TargetAccessPolicy>

  <PermissionTransferPolicy>
    <!-- Final format needs to be evaluated. -->
    <DataItemList>
      <!-- Data items on which permissions are transferred. -->
  <!-- The mechanism for transferring the permission
       needs to be determined. -->
  <DataItem ID="data-itemID1" type="urn:tas3:E-Portfolio"
            ownerref="JobSeekerTasks"/>
      <DataItem ID="data-itemID2" type="urn:tas3:CV"
            ownerref="JobSeekerTasks"/>
    </DataItemList>
      <AccessSpecification>
        <Right Type="read,write" DataItemID="data-itemID1"
          AllowedFor="PlacementProviderTasks"/>
 <!-- Allow the user assigned to PlacementProviderTasks
    read and write access on data-itemID1
(of type urn:tas3:E-Portfolio, defined above) -->
      </AccessSpecification>
  </PermissionTransferPolicy>

  <Audit>
  <!-- list of objects to be monitored -->
  </Audit>
```

```
</BusinessProcessPolicy>
```

In addition to the machine-readable policy, a human-readable policy should be created for each business process. It contains all information an end user needs so that he or she is able to assess what happens to his or her personally-identifiable information in a particular business process. A human-readable policy can be formulated in natural language as terms and conditions of the application, the business process implements. Another possibility is a representation of the security constraints related to the steps of the process execution. This can be generated automatically using the security annotations of the business-process model and logging information. In Deliverable D3.3 [3], Section 6.5 we give a hint of a auditing facility based on business processes (BPs). It visualizes data processing in real time, using the graphical process models one deploys on a BP engine for execution. This is a proposal of a user-friendly way to visualize the effects of a business-process policy (see also [62]).

Based on machine-readable and human-readable policies for business processes, we can think of a trustworthy infrastructure around business processes, as follows:

- A certification authority certifies that a business process and its security specification on the technical level comply with the human-readable policy. This policy is coupled with the executable business process using a cryptographic signature.
- In a production environment, the business process engine ensures that only certified business processes are executed.
- The user interface for business processes allows the end user to view the business process policy of certified processes and ensures that it is authentic.
- Audit log entries, collected by a trustworthy audit bus, can be traced back to the business process model that performed the action causing the log entry.
- Other parties in the trust network can determine whether a certain action is performed by a certified business process.

## 4.5 Policy management for secure business processes

In a complex trust network such as TAS$^3$, different sources of configurations of the components exist, and configuration of different sources has to be applied at the same time. As Deliverable 2.1 [23] states, TAS$^3$ “is pervasively model-driven”. In the context of business-process management this means that security configuration is derived from models of business-process applications. The purpose of this section is to clarify the relationship between the different policies and other kinds of configuration that play a role with respect to secure business processes.

The following two dimensions have to be considered here:

- There is security configuration on the modelling level and on the execution level. On the modelling level, security properties of business processes are expressed as annotations. This is described in detail in Section 4.2. These annotations are transformed to security configuration needed on the execution level (Section 4.2.4).

  This is one possible source for security configuration at runtime, as described in [1]. Other possible sources are default configurations and configuration that becomes only known at runtime, e.g., because it depends on users participating in the business process. As an example, think of a user participating in the process who selects a service provider. This selection is stored in a security-component resulting in the selected service to be called. To handle this dynamic configuration, a respective security annotation is transformed by adapting the process model with a process fragment “service selection by the user”.

- We also have to distinguish between policies that are specific to business processes, and other policies. Examples are history-based constraints such as separation of duty, on the one hand, and simple attribute-based constraints for performing tasks, on the other hand. The latter variant also plays an important role in secure business-process management, because business processes dynamically bring together different actors, including both humans and automised systems. We deal with this dimension in two ways: First, the business-process policy includes BP-specific parts, such as history-based constraints and rules for selecting identities used in outgoing calls. In addition, it includes non-history based classic access-control policies, stating, e.g., who is allowed to perform a task. These are expressed in languages like XACML or PERMIS. Evaluation of these parts uses a classical PDP. The respective (XACML or PERMIS) policies are embedded into the business-process policy; the T3-PDP-BP hands them to an appropriate general-purpose PDP (a so-called master PDP) together with the decision request. [63] describes this relationship in more detail. Second, the BPMS co-ordinates evaluation of different policies according to the entities involved in business-process instances. For example, it can discover trustworthy services to include in the business process based on the trust policies of the affected participant of the process. This can be achieved by calling a trust PDP provided as an identity web service. The BPMS could also enforce policies applicable to data processed by the business process, for example, sticky policies.

## 4.6 Securely Adapting Processes

From the perspective of business processes security is particularly important in the context of process adaptation. security rules and policies relate to processes. In case of the change of the process flow or of components of the process, we additionally need to handle the security context so, that the resulting process guarantees sufficient security. If not, the system must reject the adaptation or we must choose another one. Next, security rules define scopes of responsibility within the business process. This serves as a basis to guide business experts or even particular stakeholders of the process to adapt the process in a semi-automatic way (if automatic adaptation is not applicable).

Adapting processes needs more support for active process instances. According to the requirements identified in Section 3.2.2, we envisage the following steps to support process instance adaptations:

- choice of process alternatives or process patterns,
- change of the process schema, and
- migration of process instances.

In the following we will first give an overview about the aspects of adaptation we focus on. Then we describe structural adaptation handling in detail. There, basic mechanisms and concepts of process adaptation are introduced, on which the approaches described next rely on. The section continues with the description of several variants of how we support substitutions of parts of the process. It concludes introducing the adaptation of processes using canned process fragments. The last approach takes place in the modelling phase, changing in a controlled manner the business process schema before deployment. To configure security for business processes we focus on using process fragments, which also contain human activities and user interactions as well as calls of dedicated security web services of the secure BPMS.

### 4.6.1 Overview about adaptation of secure business processes

Our goal is to support control of the security and trust context of the business process (BP) for changes of the BP:

1. during business process modelling,
2. for choosing adequate web services, process alternatives of subprocesses, and
3. during migration of a process instance, i.e. a running business process, to the changed process, e.g., using the call of an alternative web service or subprocess.

Adaptation may concern both, process models and running process instances, and, therefore, needs to take security into account at both levels. Regarding the support of adaptation of business processes for running process instances, we have enhanced our first approach for structural adaptations. For this, we have investigated a formal model of BPMN processes adequate to take the process status into account. It also gives insights how to specify adaptations to prepare control on process change operations.

Next, we focus on security mechanisms influencing the adaptation process. To this end, selecting adaptation alternatives should use selection parameters which are based on trust and security properties of the processes, services, data and users involved, and their relationships. Additionally, in order to assess the validity of the process adaptation envisaged, we have to observe the trust and security level of the process as well as the authorisations of the process participants. To improve user centricity is an important goal for applications handling privacy-specific data, e.g., personally identifiable information. Applying BP-specific security to such applications leads to further requirements and possibilities for flexible process management. How this goal may affect the adaptation process and its mechanisms turns out to be an important factor to design and support the adaptation process.

To let process participants adapt the process would allow to improve the involvement of users. However, the process model contains a lot of knowledge about the sequences of tasks allowed, and adapting the process model is a critical task, which should not be possible in any cases. For using credentials and security rules to guide adaptation, [64] provides an authorization model for process adaptation with certain authorization constraints restricting change operations of processes.

In particular, to better involve users and use trust and security issues influencing process adaptation, we developed another approach. This approach takes security specifications of BP models, i.e., annotations of BPMN elements according to our security annotation language (see Section 4.2). Those annotations which result in a transformation into a flow of tasks to ensure their enforcement, uses a process adaptation technique, i.e., the adaptation of the process with so-called process fragments. This goes beyond selecting concrete web services (representing, e.g., subprocesses) or inserting or changing activities of a process, as provided by the fundamental adaptation approaches, we discuss first, in this chapter.

Fully automatic adaptation will be possible in special cases, e.g., adding/changing data or subprocesses to query data sources, or adaptation because of security or trust issues, e.g., that we discover a web service with the same interface but with another (higher) trust level than the preselected service . Another category of adaptation support provides semi-automatic adaptations guided by users, as, e.g., user interface of an assessor or candidate in the APL employability scenario choosing a similar service, or the interaction with the student in the Nottingham student enrollment scenario, e.g. to lower the required trust level, so that the service becomes usable. We want to provide a repository of process patterns to support the automatic and semi-automatic adaptation. An important issue to come to powerful adaptation support is to specify process semantics in order to discover adequate new tasks or subprocesses, i.e. complex activities.

#### Correctness of BP Adaptation

A main issue of adapatation is to achieve a correct process model. Properties of correct process models are, e.g., defined in petri-net based models as soundness [65]. This guarantees that process instances following this model reach the endpoint given with no side-effects like process branches remaining in an intermediate status. Process modelling tools usually provide testing of the correctness of the resulting process models to reach a sound model. We, therefore, rely on the existing correctness tests of the process modelling tool.

Further correctness criteria have to be observed if the adaptation concerns process instances. In particular, there is the need to check the data flow and the current data status of the process instance (see, e.g., [66]. E.g., if the adapted part of the process requires additional data or does not further provide data required in the old process model at the current process status would result in incorrect process flows. We have investigated such correctness criteria and our adaptation approaches introduce respective checks, see the following subsections.

#### Adaptation Approaches

In the following we introduce our different approaches for process adaptation, which are also possible to combine. For structural adaptation of business processes, in Section 4.6.2, we are focussing on the migration of process instances concerned. This is a fundamental adaptation mechanism for running process instances supporting a set of structural changes of the sequence flow.

In Section 4.6.3, we present how to manage substitutions of parts of the process regarding the semantic level of subprocesses and web services as well as the technical level of business process execution. This adaptation approach should also include mechanisms to support stakeholders of the process, defining and controlling the process adaptation. These techniques yield in more flexible business processes, and adapt business processes instances. Most of these techniques allow to determine or change web services to be called by the process.

The next approach in Section 4.6.4 handles with adaption of processes on the model layer, but allows to include security-specific flexible process fragments introducing user involvements and flows of activities including calls to components of the security framework. In Section 4.2.4 we describe our main technique for configuring business processes with security. To this end, we are using business process adaptation for transforming security annotations of business process models.

### 4.6.2 Changing the Process Structure

We developed an adaptation concept with the objective to get able to perform structural modifications on a BPMN model and to convert these changes to running instances of this model.

The challenge was to develop such a concept that structural modifications on running process instances can be performed on the BPMN layer, inside the graphical BPMN modelling tool. So a business analyst or a process participant is able to execute changes on running process instances without the need to have advanced skills in business process management on the technical level.

”Structural modifications” hereby means changes on the process model that affect directly the process flow, for instance the insertion of a new activity or the deletion of an existing one.

#### Challenges

There are several publications on structural adaption of running processes. At most, these approaches are based on process models specifically developed for this purpose, therefore their practical applicability is relatively restricted. We took the work of Manfred Reichert [66], which is based on an adaption-specific process model, as a landmark to develop an adaption model based on the industry and OMG standards BPMN and BPEL.

We focused on designing an integrated adaption concept containing all affected process layers: the BPMN layer, where the process is graphically developed with a BPMN modelling tool, the process definition layer with BPEL and the instance execution layer in a business process execution engine (see Figure 4.19).

An essential challenge was to comply with integrity and consistency constraints, in particular on the instance execution layer. First of all, we have to guarantee the correctness of the process models on the BPMN and on the BPEL layer after process modification. On the instance layer, the maintenance of instance execution state integrity and the maintenance of data flow correctness are crucial. We will examine these constraints in further detail in section 4.6.2.

Two fundamental obstacles hindered the development of an integrated adaption model. First, there exists no formal meta model for BPMN, so far. Today BPMN is a model on the graphical level only, deposited with attributes. Version 2.0 of OMG’s BPMN standard will define a formal meta model, but this version is still under development. On the instance level there does not exist a formal model for instances of BPEL processes either. At least there is a meta model for BPEL defined as a XML-Schema document available at the BPEL 2.0 specification. To define a model for the adaption of BPEL instances modelled with BPMN we need meta models for all affected layers. So we had to develop models for BPMN and the

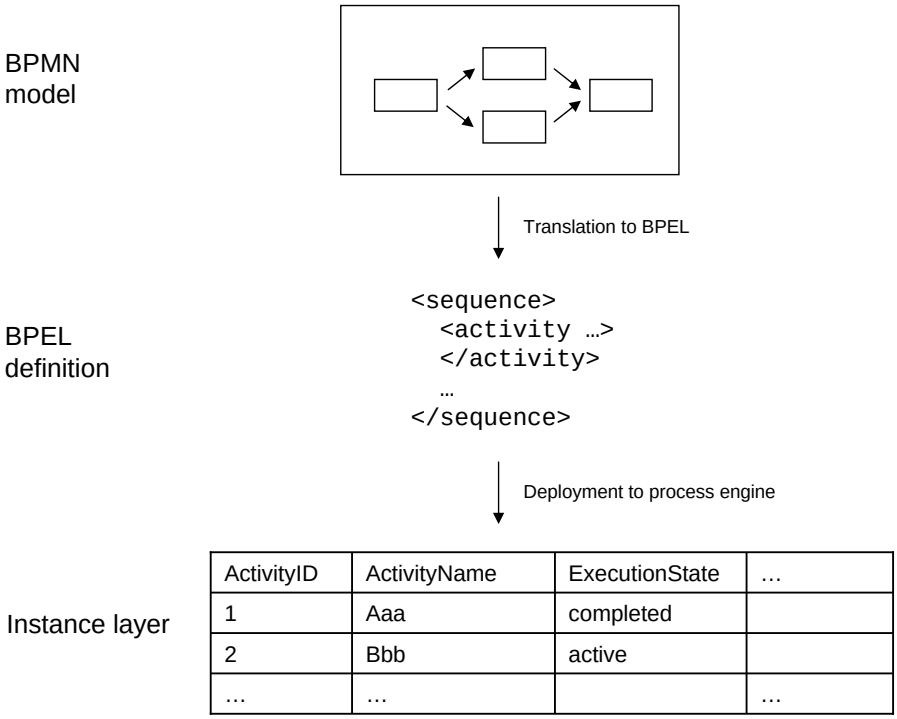


**Figure 4.19: Three layer model)**

BPEL process instances.

Secondly, there is still no standardised approach for translating BPMN to BPEL. The BPMN 1.1 standard [75] contains a non-normativ proposal of how to map BPMN 1.1 process models to BPEL 1.1 [76]. But to the best of our knowledge there exists nothing comparable for the translation of BPMN 1.1/1.2 to the current version of the BPEL standard, BPEL 2.0. The fact that BPMN to BPEL mapping is still an open topic is caused by the differing expressive power of the two meta models. Therefore, a full mapping of BPMN to BPEL is not possible in general. For our adaption approach we do not need a full mapping of all BPMN elements and structures to BPEL, but a solution which covers the essential ones. For this purpose we reverted on the translation approach proposed in the BPMN 1.1 standard and adopted it to BPEL 2.0.

#### Integrity and consistency constraints

In the following we will give an insight into the integrity and consistency constraints every process adaption concept for BPMN and BPEL has to comply with.

**a) Process model correctness** Within the scope of an adaptation concept for BPMN and BPEL we have to deal with two process model layers. Modifications on one layer must affect the other one. In our user-oriented approach we assume that changes of the process are performed within a BPMN modelling tool, i.e. on the BPMN layer. Depending on the adaptation concept these changes have to be transferred to the corresponding BPEL process model. In our adaptation model we do not adapt the BPEL process definition manually but make a common translation of the BPMN model by an available BPEL compiler.

**b) Execution state consistency** Regarding the maintenance of the instance execution state integrity, two aspects are essential. The first question is in which execution states of a process instance modifications on the underlying process model are allowed. It seems to be obvious to allow only modifications on process sections that have not yet been executed. But regarding to process loops or compensations it might be reasonable to allow modifications on selected parts of the process that have already been executed on

the instance layer.

It can be necessary to adapt the execution state of a process instance, whose process model has been modified. This is because there is no one-to-one correspondence between BPMN and BPEL elements. We give a short example for this.

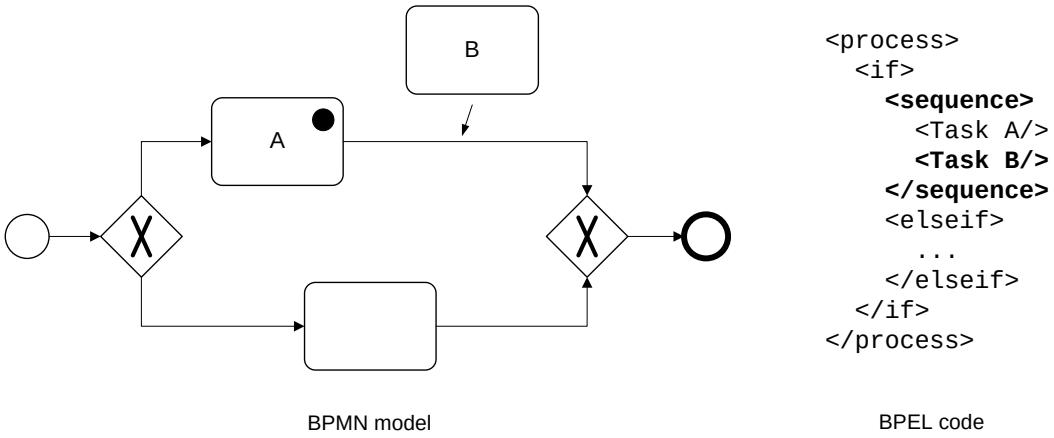


**Figure 4.20: Insertion of an activity causes creation of an additional structured activity**

Figure 4.20 shows a BPMN process model and the corresponding BPEL process definition. We want to insert a new activity called ”B” into the process. On the BPEL layer this causes the creation of a basic activity called ”B”. In addition to that there is the need to create a new complex *sequence* activity in order to comprise the basic activities A and B.

Assuming that, on the instance layer, activity A is currently being active, it is necessary to assign the status ”active” to the complex *sequence* element, too. Otherwise the instance execution status would become inconsistent.

This example shows that we have to adopt not only the execution state of modified activities to the general execution state of the instance, but also the state of other BPEL elements may be affected by the modification operation. This might even affect additional BPEL elements, that do not have a corresponding element on the BPMN layer.

**c) Data flow correctness** The correctness of data flow is another important issue in the development of process models and in the adaptation of process instances. Surprisingly, neither the BPMN specification nor the BPEL standard define correctness criteria for data flow correctness. As the only declaration of data flow correctness made in BPEL is the definition of a *bpel:uninitializedVariable*-Fault, which is thrown, when trying to read a process variable, that has not yet been initialised.

This approach may be sufficient for the development of process models where the data flow correctness can be quite intuitively verified on the graphical layer. But this no longer holds for the development of an adaptation concept. Dealing with running process instances means that there may occur side-effects of the process modification operation that are not visible on the graphical layer nor on the BPEL layer, but could have considerable effect on data flow consistency of the affected instances.

Figure 4.21 shows an example. The correct process in the upper half of the figure, contains a write-read sequence. Both activities access variable A – the data flow is correct. After modifying the process model in that way that both, the write activity and the read activity use variable B, the data flow of the modified process model is totally correct, as well.

But on the instance layer a problem might occur: Imagine an instance of the process in which the write-activity has already been completed and the read-activity is not activated yet. This execution state shows the underpart of Figure 4.21. When migrating this process instance to the new process model, then a data flow problem occurs. The write activity has been executed before adaptation with write access to variable A, and the following read-activity (running on the new process model) expects variable B to be written.

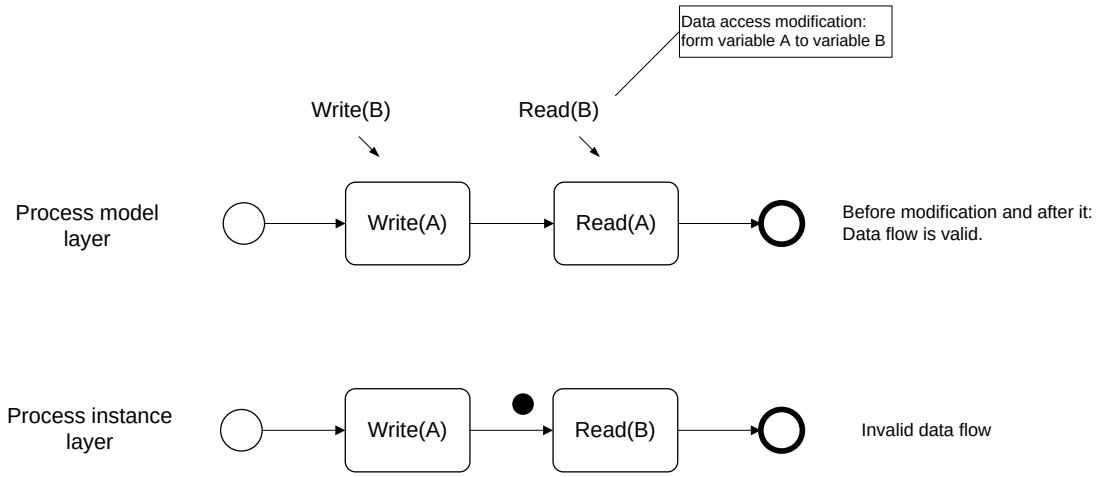


**Figure 4.21: Differences in data flow correctness on the process model layer and on the instance layer**

This example shows that we have to deal with a different understanding of data flow correctness on the process model definition layer and on the instance layer. The degree of relevance of this aspect to an adaptation concept is highly dependent on the question if modifications of already completed parts of the process are allowed or not.

#### Adaptation concept

Figure 4.22 shows the general approach of our process adaptation concept. We come from the idea that the process administrator is able to apply a set of well-defined adaptation operations to an existing BPMN process model. He or she chooses one operation, e.g. *TaskSerialInsert*, and provides the necessary parameter values. For simplification reasons we do not allow free-hand modifications on the process model, so far.

A common BPEL compiler then automatically translates the modified and validity checked BPMN process model to BPEL representation. Using this representation we have to migrate the running process instances. Depending on the adaptation operation performed, several validity and integrity verification operations must check the correctness of the modification, before finally migrating the process instance to the new process model.

These steps concern mainly the data flow verification and the inspection and possibly the adaptation of the instance's execution state. We discussed these aspects briefly in section 4.6.2. For the verification procedure in complete we refer to the description in [77].

We also developed meta models for a relevant subset of BPMN and for the instance layer to define the data flow operations, the verification step of the process execution state, and the instance migration, itself. The BPMN meta model is a formal version of the BPMN model described in the BPMN 1.1 standard. The definition is available in [77].

Our instance layer meta model consists of the relevant part of the BPEL process model, the current instance execution state and the current variable values, see Figure 4.23.

As already discussed before, for defining an adaptation model it is essential to provide a well-defined model for the translation of BPMN to BPEL. We split the top-level *adaption operation* into an adequate *BPMN modification operation* and an appropriate *instance adaption operation*. The details of the *instance adaption operation* (including data flow verification and instance execution state adjustment) can only be specified if it is formally defined, how the mapping of the BPMN process to BPEL looks like. We made some restrictions. First we focus on a set of basic adaptation operations. Further, we assumed that we have a block structure of the process models, which is valid for the BPMN part handled with some further restrictions. In particular, on the BPMN layer this means, e.g., that a splitting gateway must be followed

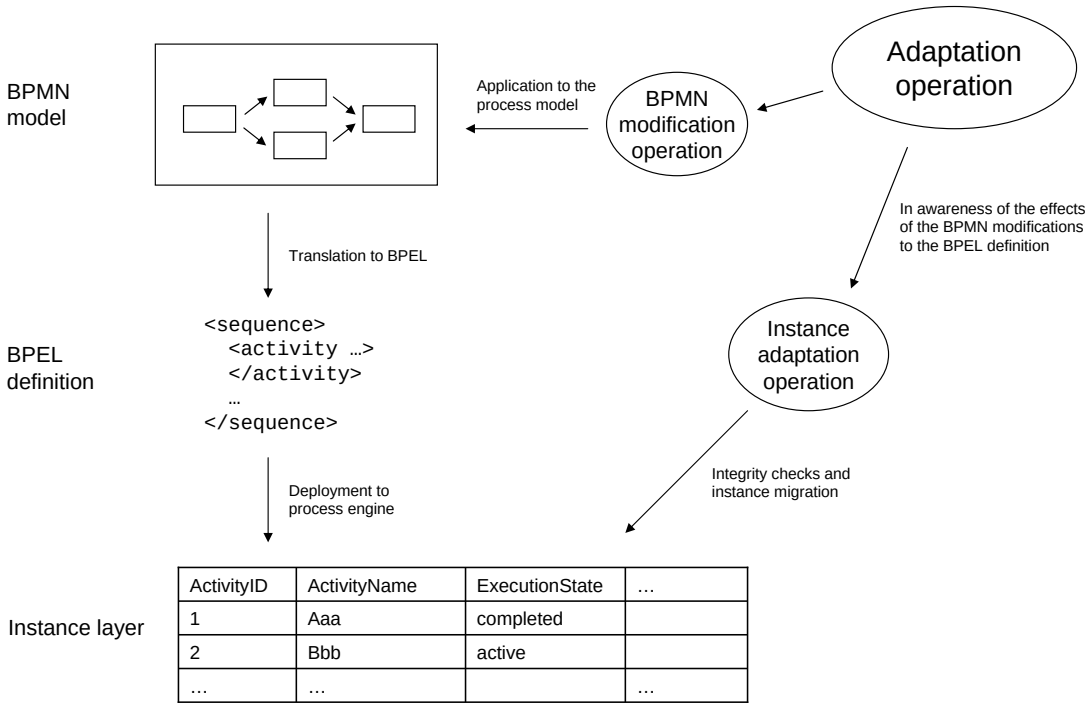


**Figure 4.22: Adaptation concept**

Process instance := process definition + instance execution state + instance variable values

**Figure 4.23: Meta model of a process instance**

by a merging gateway. On the BPEL layer we do not permit using *flow* activities in general. For further enhancements a formal and standardised BPMN 1.1 to BPEL 2.0 mapping of the full models becomes an essential requirement.

Section 5.8 contains the architecture developed for our adaptation approach.

### 4.6.3 Substitution of Parts of Business Processes

In the requirements section (Section 3.2), we have already mentioned that substituting web services or subprocesses during execution of a business process is not sufficiently supported by existing business process execution engines. Using abstract web services during design allows to exactly determine one concrete web service for execution before its call. To this end, we have to model the determination of the web service explicitly as activity in the business process itself. This is also supported by the adaptation technique using process fragments.

To become more flexible we are proposing methods on different levels which may be combined.

On the technical level, a first possible way is to allow for substituting a web service call by a subprocess, which is defined internally in the business process execution engine, i.e., the business process engine, in our case the Apache ODE which is also the execution engine of the Intalio/BPMS, has to be modified. A first naive approach is to substitute each web service call by such a subprocess. This may result in too much overhead during business process execution.

Meeting this problem needs support on the business process modelling level. We will need a methodology to determine those subprocesses and web services that should be prepared for dynamically replacing

them during execution. In literature there exist other proposals like [72] which introduces a process meta model with parts, i.e. activities, that are modelled as abstract web services and replaced in an automatic manner by searching a web service with a discovery tool. The drawback is the meta model which restricts the places in the process where substitution is possible. We will go another way: starting with a multi-level business process modelling methodology like the PMF, see Chapter 3.1.1, the process designer can define a level starting from there down the hierarchy all childs in the subprocess hierarchy are realized as abstract web services. Explicitly defining a subprocess or web service as fixed will stop the use of abstract web services in that branch in the child nodes.

The next step aims modelling the internal substituting subprocess. We are providing a generalized subprocess with parameters to define the endpoints of the subprocess or web service as minimum parameters. We provide the following specializations of this subprocess:

- The direct call with parameters of the endpoint reference of a web service.
- Using a discovery service which searches for adequate web services.
- Using a specific repository, e.g. local to the business process engine, which allows at modelling level to define alternative subprocesses and patterns and store them semantically annotated in a semantic web service repository.
- Using a user web interface which interacts with the business process actor and gets information about the choice of the concrete web service with help of the user. This is a flexible way, to enhance user influence, but it needs further preparation, e.g. providing a list of possible web services or subprocesses with properties, e.g. the trust level, assisting the user.

#### Direct call

The first variant is a basic implementation of the internal substituting subprocess, we have realized it as first approach and basis for the other variants. Figure 4.24 shows its BPMN representation.

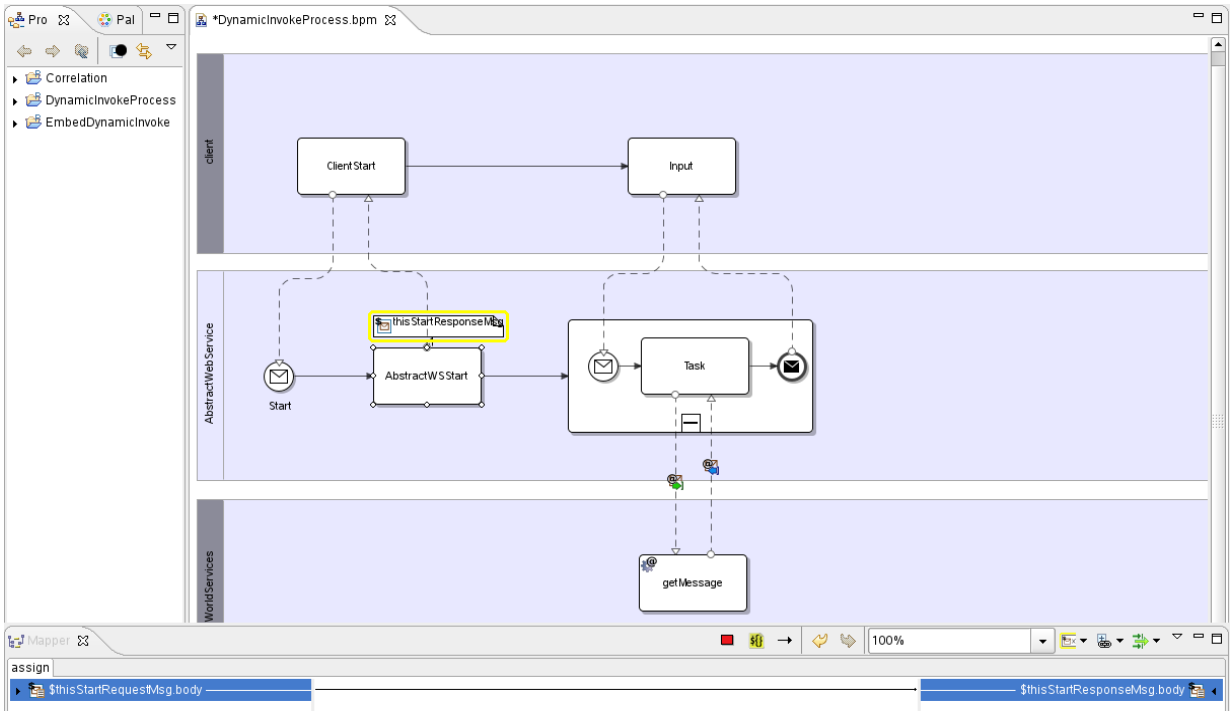


**Figure 4.24: BPMN model of the internal subprocess instantiating an abstract web service.**

#### Service for service discovery

The second variant needs a discovery service with web service catalog as possibly available in a service-oriented framework. In our case we require security and trust properties for discovery, e.g. calling a trust and security negotiation service, which is part of the TAS$^3$ security and trust framework. To this end, we are using these discovery mechanisms and are changing the web service calls of a business process in that way, that they are using service discovery calls and calls to the Policy Enforcement Point (PEP) of the business process management system.

### 4.6.4 Changing Processes Using Process Fragments

A possibility for process change is to enhance the process model by business process fragments from a repository. These fragments need to be described semantically. There exist several approaches to use so-called semantic web services which are provided by semantic repositories. Known approaches are OWL [67] and OWL-S[31] [68], WSMO [69] [32], and WSML [70] as ontology based descriptions of web services and matching functionality to find adequate web services in a specific situation.

If we enhance this technique to business processes, the starting point is to see subprocesses as web services. But we want to adapt existing processes with these subprocesses. There are proposals to compose web services to more comprehensive ones. The application of these techniques in practical application is still limited, because mostly a well-defined environment is a pre-condition for providing connectable web services and to have repositories with usually application-dependant services.

In our scenario, we are working in a heterogenous and open service-oriented framework, but our goal is to use adaptation for the purpose of adding security-, privacy- and trust-related functionality. With that, we can work in a well-defined area to be able to provide a repository and to use the contained fragments for security, privacy and trust purposes.

The following steps are provided:

- defining the structure of the process fragments
- define the description of the process fragments
- provide an architecture and mechanism to insert process fragments in existing process models in an automatic way

#### 4.6.4.1 Definition of Process Structure

To realize security-related functionality, the process modeller has to define the structure of process fragments. The structure depends on the security annotation. We realize any user constraints that refer to security, privacy, and trust issues (see Section 4.2.2 and the Appendix 6) by process fragments. This means that parts of processes for any kind of user interactions have to be modelled and maintained in a repository.

For example, the annotation
$\ll UInvolve:$
$type =$“$Consent$”
$receiver =$“$TAS3PrivacyTerms$”
$role =$“$Learner$”
$display =$“$yes, no$” $\gg$
of the Nottingham Student Placement Process will be represented as process fragment with the following tasks: “send”, “display”, “answer”, “receive”. The first and the last task are in the responsability of the placement process (to control the process), and the tasks “display” and “answer” are user interaction tasks asking a learner by an interface to agree on the consent. Additionally, exception handling has to be considered for the case that a learner does not agree on the consent (e.g., by means of terminating the process).

#### 4.6.4.2 Description of Process Fragments

To perform process fragments that are stored in a repository, they have to be described semantically. The description relates to the security annotation and the meta data include the category of the annotations and other parameters of the annotation, e.g., the type of a UInvolve annotation.

For example, the annotated ”UInvolv” above can be described as follows (simplified):

```
<ProcessPattern>
  <!-- Description of the Process Pattern . -->
  <PatternType> UInvolve Type </PatternType>
  <Pattern> UInvolve
   id="...."
   type= "Consent"
   receiver="TAS3PrivacyTerms"
   role="Learner"
   display="yes,no"
  </Pattern>
</ProcessPattern>
```

The attributes “type”, “receiver”, “role”, and “display” will be used during process execution as process fragment parameters that are used to concretely define the relationships between the data objects of the process pattern and the adapted process. E.g., they provide parameters to perform user interactions contained in the process fragment.

#### 4.6.4.3 Implementation

We have realized such an adaptation mechanism for security annotations (see [71] and Section 5 of Deliverable [**?**]). With that, we have investigated the three steps, implemented a prototype and validated the mechanism by using it in two e-employability demonstrators, the second Nottingham student placement business process application and the Kenteq mass layoff application (see Deliverable citeD3.3.3,Sections 3 and 4).

## 4.7 Security of Business Processes

Our approach to facilitate the security of business processes is to build on well-established solutions to well-known problems. For example, the concept of roles and role-based access control (RBAC) is based on the insight that there is always some classification of users, and that it applies to access control decisions as well. We extend the basic RBAC approach by introducing new abstractions tailored to the domain of processes, taking into account that processes glue together the resources and actors involved. We do not propose security settings to be tailored to a specific run-time scenario, making them less generic. Instead, we use the composition and execution context (i.e., which actors and resources are involved, and which activities are currently executing) of running process instances to let the system make access-control decisions. This means that we strive to include security parameters in process models in a declarative fashion wherever possible. This is not always sufficient, and we supplement it by enriching process models with explicit (imperative) actions, such as the assignment of individuals to process roles. The partitioning of processes into several layers of sub-processes, as part of the process-modelling framework (PMF), makes process diagrams easier to read and prevents issues with only local significance from cluttering up the global view. In line with this approach, we plan to allow policies that are specific to sub-processes.

The solutions presented in this section are structured in the same way as the requirements in Section 3.2.1. Section 4.7.1 addresses authentication and identity management. Sections 4.7.2 through 4.7.5 address authorization. [26] also summarizes our findings of requirements on security for business processes and their impact on advanced security mechanisms and serves as basis for the conceptual design in this

chapter. We introduce a formal model for the execution of business processes in Section 4.7.6 and use it to formalize security concepts in 4.7.7. T, the business-process specific security concepts become more clearly specified providing a fundamental description for their use and implementation, and make it more understandable that we get a richer variety of security support for SOA applications that are based on business processes compared with those directly programmed without explicit representation of the sequence flow and its interactions with actors, resources and external events.

We will look at the run-time behaviour of business processes in the implementation chapter, namely sections 5.7 and 5.8. In Chapter 5, we will also present preliminary solutions for the cross-cutting concern of modelling. The order of the solutions in this chapter follows that of the respective requirements.

### 4.7.1 Federated identity and single sign-on for the user interface

Business processes in a distributed service-oriented environment include actors from several different organisations. They interact with the business process via a task-list console. In order to do so, they must be authenticated, i.e., they need to log in.

The current APL process as installed at Kenteq is a hard-coded application with additional tasks performed manually. When the TAS$^3$ infrastructure will be in production, the Kenteq process should run as a business process in the secure infrastructure. When the APL process accesses PII of the user from different services, ist must use the federated identity system of TAS$^3$. the user is known by a different pseudonym those services. The business process must use the Identity Mapper to translate the user's IM token (Id Mapper bootstrap token) to a token usable for the web service that is about to be called.

Further, single sign-on based on federated identity must be supported by the task-list console so that users do not need to keep separate logins for different service providers in the TAS$^3$ infrastructure.

### 4.7.2 Instance-specific user-task assignment

In the design of a business processes, it will become apparent that several activities logically belong together and should be assigned to the same resource. These groups of tasks can be seen as roles. They group responsibility for performing activities in a business process. This distinguishes them from roles as used in role-based access control, which are used as an abstractions to group permissions. We need to carefully distinguish both uses. Thus, we use the term "group of tasks" when referring to the former.

In business-process-management systems, roles are used for access control as well as for task assignment. The access control aspect is about coupling permissions to roles and roles to individuals, thus making permission assignment easier and more flexible. In the next paragraph, we briefly describe the Role Based Access Control (RBAC) model. We also describe attribute-based access control. Then we will explain how to use these models for user-task assignment in business processes.

Role-Based Access Control is a mature access control model. It has first been described in 1992 [78]. In 1996, Sandhu et al. proposed their own version of RBAC [79], leading to a unified standardization proposal in 2000 [80]. The result was a formal standard published in 2004 [81]. RBAC is a general-purpose access control model, not specifically tailored to business process management. The standard only defines the relevant entities (user, role, session, permission) and their relationships. It explicitly does not propose a machine-readable representation, scalability requirements, and the like.

The RBAC reference model is defined as a collection of four components: Core RBAC, Hierarchical RBAC, Static Separation of Duty Relations, and Dynamic Separation of Duty Relations. Core RBAC includes users, roles, permissions and sessions. Permissions are assigned to roles, and users are assigned to one or more roles. Users activate roles in the context of a session and then hold the permissions assigned to these roles for the session. Hierarchical RBAC adds a role hierarchy. The remaining two components define relations between (conflicting) roles. Static separation of duty puts constraints on the assignment of roles to users, while dynamic separation of duty limits the simultaneous activation of roles by a user in one session. We will treat Separation of Duty (SoD) in more detail below.

An obvious application of RBAC to business processes is the interpretation of sessions as business-process instances. Dynamic separation of duty then results in a separation of duty between roles over one process instance. This means that the activation of a role by a user is valid for an entire business-process

instance. However, organization-wide roles are not sufficient to support versatile business processes: Similar (or similarly named) roles in different process models can have very different prerequisites. In other words, there is not necessarily a one-to-one match between overall organizational roles and the roles in business process models.

We briefly recall requirements relevant to the handling of roles in business processes: We need process roles that are separate from organisational roles specified elsewhere, and we want to be able to explicitly assign individuals to roles (requirement D3.1-R.3). On the other hand, we still want to be able to re-use existing organisational roles in a business process (requirement D3.1-R.4). Further, it shall be possible to specify that a role implies binding of duty, i.e., that the same person executes all tasks for the role (requirement D3.1-R.6).

The RBAC model does not specifically capture business processes and process instances. Interpreting sessions as business process instances solves the problem only partly: RBAC does not support constraints on the activation of a role for a business process instance that take the characteristics of that instance into account. This means that a user can log in and activate a certain role in any process instance, without any guarantee that he actually has the necessary skills.

In attribute-based access control [82], the more fine-grained concept of attributes replaces roles. This is actually an refinement of RBAC, because roles can also be expressed as attributes.

To fulfil the requirements and to overcome the shortcomings of the RBAC approach in our specific context, we use groups of tasks. The assignment of actors to such groups is stored separately for each process instance. It is subject to a policy that specifies criteria which the user assigned must fulfil, based on attributes of the user. Such a policy is defined for each task group in the process model. Role mapping (requirement D3.1-R.4) is a case that does not need any special treatment, because "role" is an attribute of users like any other.

Different entities take assignment decisions at different points in time: Some assignments are already fixed when the process instance starts and passed to it in the start event. In other cases, the process itself (by an explicit activity) or a user performs the assignment when the process is already running. In most cases, a user holding a special role will be charged with such decisions.

In some cases, it is also possible to let the execution engine choose an individual for a group of tasks. Different strategies are conceivable as long as the assignment policy for the role in question is fulfilled. The system can then choose either an eligible actor itself and assign the tasks to him, or it can present the first task for the role to all actors eligible and choose the first individual who claims the task.

In Section 4.7.5, we introduce constraints on user-task assignment such as binding of duty and separation of duty.

WS-BPEL processes have a semantics based on their compositional structure. It is desirable to use this structure in rules for user-task assignment. This means that the definition of task groups and corresponding user-task-assignment rules will be attached to structured activities (such as a $< while >$ loop in a WS-BPEL business process). They will then only be valid for activities nested within the structured activity. For loops, this means that the specification applies to iterations of the loop seperately. This requires an execution semantics for business processes, as explained further in Section 4.7.6.

Figure 4.25 demonstrates what happens when the business process performs an explicit assignment of a user to a group of tasks. The process, based on the user decision, calls the BP-PIP to perform the assignment. This request passes through the PEP which adds information to identify the business process model and instance. The BP-PIP hands the request on to the PDP-BP. The PDP-BP fetches the assignment policy for the task group in question checks the assignment. It then communicates its decision back to the BP-PIP. Assuming that the assignment is allowed, the BP-PIP records the assignment. It sends a reply to the business indicating success. Additionally, it notifies the permission management components about the change in the role assignment. If the assignment is not allowed, the BP-PIP will not record it. The reply to the business process will cause the PEP to raise a fault. The process must handle this fault to avoid abrupt termination.

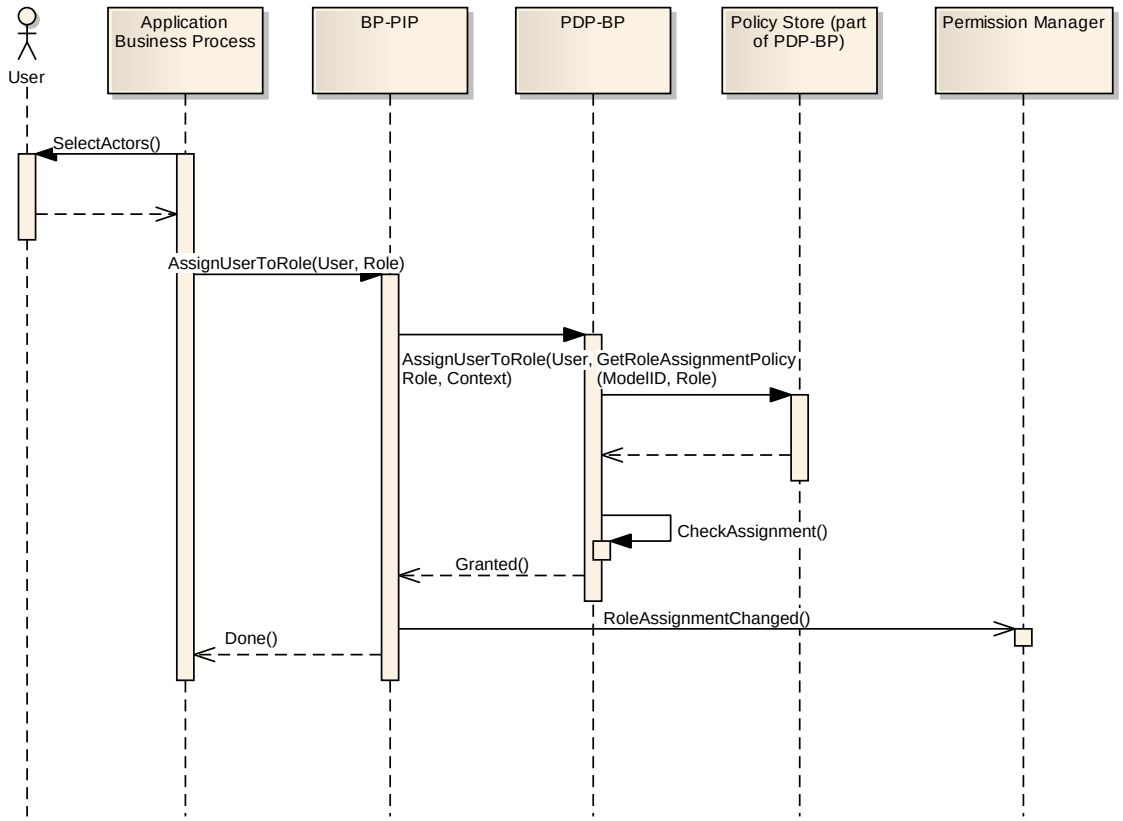


**Figure 4.25: UML Sequence diagram demonstrating the conceptual message flow on an explicit user-task assignment**

### 4.7.3 Permission Management Aware of Business-Process Context

As stated in requirement D3.1-R.5, it must be possible to specify the resources which actors of the process may access. Further, it must be possible to specify when access is allowed and when not (or only in a limited way, e.g., read-only). There already are some approaches in the literature that aim to fulfil this and similar requirements, i.e., to limit the amount of data that a person can access and the time interval when such an access is possible, according to the context. We will briefly review these approaches and assess their suitability for our scenario. Then we describe our concepts for active security and the implementation planned.

Team-Based Access Control was originally proposed by Thomas in 1997 [83] and refined and extended by him and others in 2001 [84]. TMAC addresses collaborative environments in general, not specifically business-process-management systems. The basic idea is to capture the users and resources that participate in a task, and to limit permissions by defining them using a combination of roles and resource types. As this approach is well-suited for business processes, we propose to adopt it in TAS$^3$. However, in TMAC the users and objects involved in a team are just sets. Differences between the permissions of team members only arise from the roles they already possess outside of the context of the team, but there is no explicit relationship between abstract users (i.e., process roles) or abstract resources defined in the model and the individual users and resources assigned at run-time. TMAC assigns resources to cases. It does not take tasks into account, so it cannot grant a permission just to a person performing a certain task. This is not sufficient for business processes with well-defined roles played by individual users. We propose to address this deficiency in TAS$^3$.

Atluri and Huang [85] were the first to address synchronization between the control flow of a business

process and the authorisation on processed data. However, apart from this basic feature, their results are not directly applicable to our scenario, as they model business processes in a very different way. WAM cannot express complex control flow using branches or loops, which are possible in WS-BPEL. It specifies the types of objects used and created in a task. Authorization templates refer to object types, not to specific objects. Thus, WAM does not consider the relationship between a data object and the workflow as a whole. In addition, their model, WAM, directly specifies the subjects (users or programs) that have to perform a task, instead of specifying requirements for task allocation.

[86] presents an approach called Task-Based Authorization Controls (TBAC). It is based on *authorization steps*. Authorized users activate these steps by performing a task in a workflow, and other users gain certain pre-defined permissions. Further, the authors specify the possible states of authorization steps (e.g., 'dormant', 'valid', or 'hold') and the transitions between these states. An extension of the basic TBAC model supports composite authorizations, requiring more than one person to grant an authorization. Another extension adds constraints. Authorizations in TBAC directly refer to subjects and objects. TBAC does not derive the actual subjects and objects from the workflow context, so it does not fit our requirements.

From TMAC we borrow the concept of specifying policies in an abstract way. That means that policies are based on roles and named abstract resources. Abstract resources can be looked at as a kind of "role" for resources involved in a business process. This fits our current demonstrator process that deals with a number of distinct documents. It would also be possible to define policies based on resource types or sensitivity levels, given more detailed specifications of the data involved. In order to "instantiate" those abstract permissions at run-time, the security infrastructure needs to know the assignment of persons to roles, the matching between permissions delegated to the process, and the named resource types needs to be known. Thus, the process will be able to explicitly assign a permission to a resource so that it belongs to the process and matches an abstract resource specified by its name. At a later stage, we envision this assignment to happen automatically at the BPEL activity where the process retrieves the location of the resource, e.g., when it receives the response from a discovery service.

To be more precise, the abstract policies in our case are in the form of re-delegation rules. These rules contain the name of an abstract resource and the name of a process role. They can contain one or more intervals during which the permission is valid. At run-time, such a rule results in the re-delegation of the permission stored for an abstract resource to the user holding the specified process role. Additionally, they can contain other restrictions on the delegation. For example, the re-delegation of a read/write permission can be restricted to "read-only". The interval boundaries are stated in terms of business process activities and can either be inclusive or exclusive. This means that open, closed and semi-open intervals are possible. At a later stage, we want to check whether the end is reachable from the start, and determine (in advance) alternative end boundaries in case of, say, abrupt termination.

Specifying the validity of delegated permissions based on the execution of tasks in a business process and determining the validity based on such specifications is closely coupled to a formal execution semantics capturing the structural and behavioral aspect. We will look at this aspect in more detail in Section 4.7.6. In Section 5.4, we design an interval monitor for the open-source BPEL engine ODE. There, we also explore what kind of intervals can be determined with that engine.

An implementation has to fulfill the following tasks:

- The process policy must express (in an abstract fashion) which resources the business process (or humans/service provider participating in it) needs to access.
- The resources must be discovered/determined at runtime. The component responsible is called process permission manager (PPM) in the following.
- The data owner must be able to delegate the necessary permissions, using a trusted user interface that can cause the delegations on his behalf.
- A permission management component must be able to determine which resources are used in a process and which permission on them need to be granted to which human or service provider.

- When a permission is activated or deactivated, this change must become effective for access control decisions at the resource.

The policy presented in Section 4.4 specifies the resource types necessary in a specific business process, and contains rules describing the permissions resulting at runtime, see also [87] for permission handling in a BPMS. Regarding the actual transfer of permissions, we have explored several conceptual architectures:

- Policy-based policy administration: This architecture on features that will be available in the administrative policy profile of XACML v3[2] and its administrative policy profile [89]. The resource has an administrative policy allowing the data owner to delegate permissions on it. The PPM informs the dashboard about the resources the business process will need during its execution. After it has acquired consent from the data owner, the dashboard registers a policy on his/her behalf. This policy allows the PPM to delegate all the permissions it has received. Thus, the PPM can issue and revoke permissions according to the specifications in the process policy. This solution has a high overhead for policy administration.
- Manual policy administration: Without XACML v3 and Policy-based policy administration, the dashboard needs to be trusted by the resources and would directly access their policy repositories. The overhead for policy administration is even higher than for the solution mentioned before.
- Special attributes: In this variant, the PPM also informs the dashboard about the resources the business process will need during its execution. The dashboard registers a policy allowing access to anyone possessing a special attribute issued by the PPM. Thus, the PPM acts as an attribute authority. Preferably, attributes are provided in *pull* mode, so that the process does not have send updates every time a permission is activated or deactivated. This solution is much more efficient, but providing the attributes is challenging.
- Token-based approach: This variant depends on delegation tokens issued to actors who need permissions in the context of a business process.

We have started implementation design of the second and third solution. We choose the variant to actually implement based on technological feasibility. In the project framework both solutions are used for certain tasks.

The following figures show some prototypical sequence diagrams. The first diagram uses policy-based policy administration, while the second one uses delegation tokens.

In Figure 4.26 we demonstrate how re-delegation works at run-time. A new instance of the business process starts execution. Then, it calls the IR-PIP to assign Bob to the Coach role. The check of the assignment is omitted here. The IR-PIP informs the Process Permission Manager (PPM) about the new assignment. The PPM inquires the Policy Store about permission assignment rules involving the Coach role. It retrieves one such rule, and uses permissions assigned to the named resource slot given in that rule. As no permission is assigned yet, it ignores the role assignment. Then, Jim delegates read/write access to Jim's portfolio to the process. The process calls the PPM to register the delegated permission as belonging to the resource slot with the name portfolio. There is one matching rule in the policy: "Re-delegate read-only access on portfolio to Coach." The PPM inquires the IR-PIP about the current assignment to the Coach role and receives the reply "Bob". It causes delegation of the permission on Jim's portfolio to Bob, restricted to read-only access.

Figure 4.27 shows another example. Bob has been delegated permissions on detailed job records of Jim, but only for the interval starting with the request to write a report about it, and ending with the submission of the report. Jim accesses the repository containing the detailed job records through an appropriate interface. He uses the delegation token that he received from the process to do so. The repository's PEP uses the CVS to check the token. The CVS determines that Jim has permission to access the resource, but under the constraint that a certain interval in the process, say [A,B), is active. The PEP then asks the PDP

[2]XACML version 3 is currently under development and available as a committee specification [88]

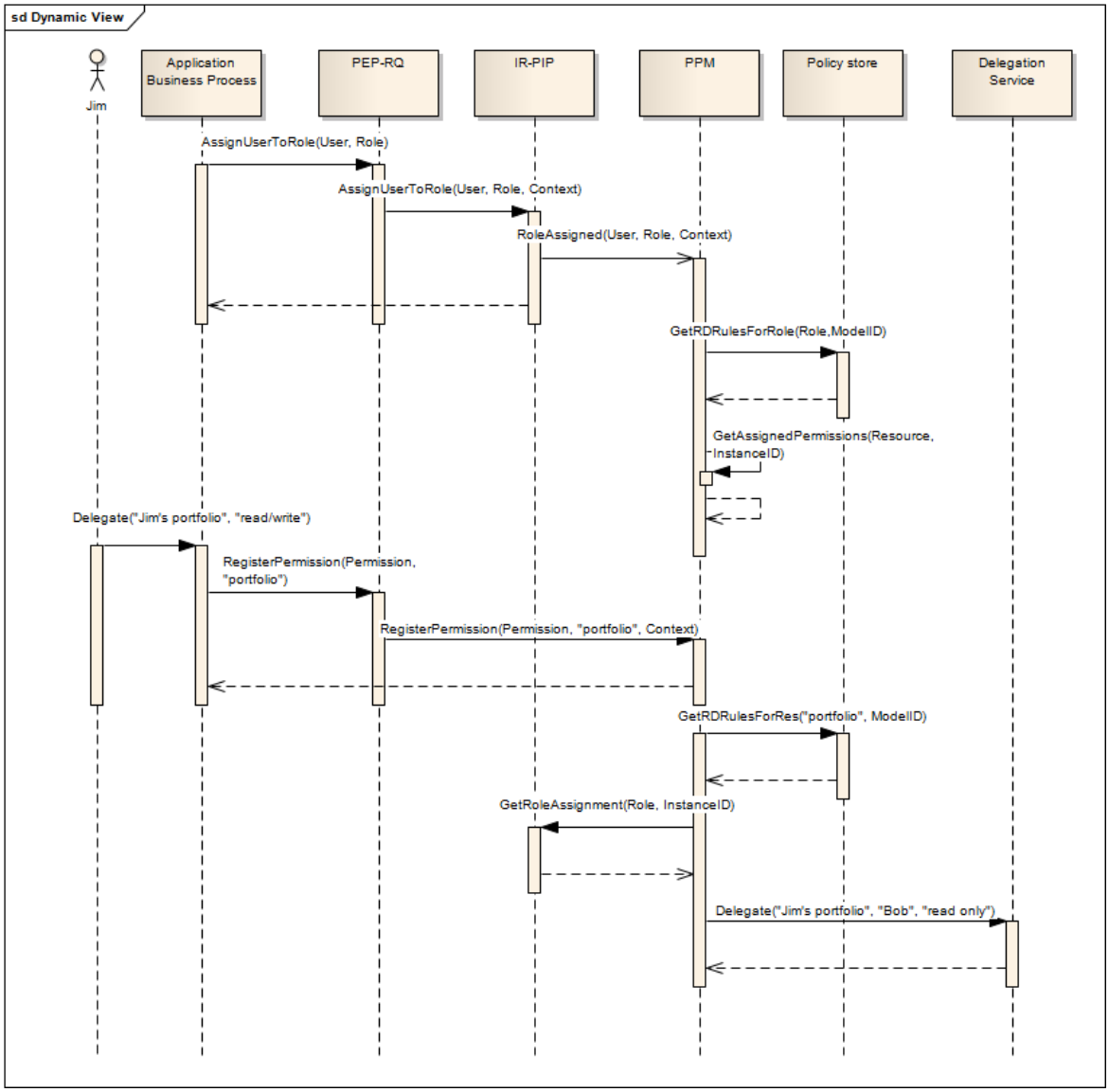


**Figure 4.26: UML sequence diagram showing the re-delegation of a permission to a process participant**

to evaluate the credentials verified by the CVS. The PDP determines that access is granted in principle, and sends a request to the Interval Monitor to determine whether the interval is active. The Interval Monitor, based on its internal state, confirms this.

Note that the flows presented in Figure 4.26 and Figure 4.27 are conceptual. A naive implementation of the Re-delegation Handler in the way it is presented here will likely be prohibitively expensive.

As briefly addressed above, there are different approaches for actually transferring permissions, inter alia:

- Changing policies or deploying and undeploying additional policies.
- Using an additional policy and special attributes.
- Using delegation tokens. One solution here would be the OAuth protocol [90]. It specifies how an authorization server can grant access to a resource by issuing an access token. This may require the resource owner to give explicit consent. OAuth itself does not provide any method for limiting the scope of access rights granted. OAuth is only defined for HTTP, not for SOAP. This limits its use for resource access by services.

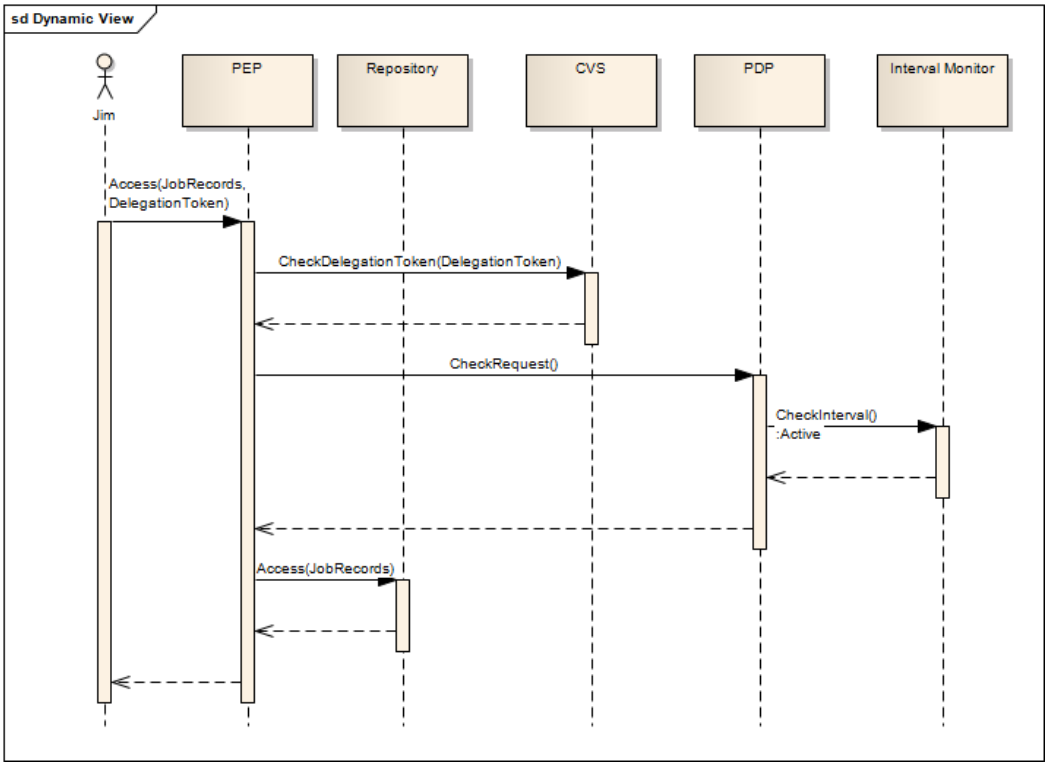


**Figure 4.27: UML sequence diagram showing how a delegated permission with an interval constraint is used**

### 4.7.4 Delegation

As stated in requirement D3.1-R.7, it must be possible to specify delegation of instance-specific roles based on particular business process instances. This way of specifying the scope of a delegation is specific to business process management. To accomplish this, the system has to enhance the dashboard to show the process instances a person is involved in. The user can then choose one or more instances and a person he wants to delegate his duties in these instances to.

Then the delegation policy is checked to determine whether the delegation is allowed. A delegation policy contains pairs of conditions, referring to the delegator and to the delegate. The conditions can, e.g., refer to their roles (role based delegation). Optionally, the policy contains a maximum length of the delegation chain. A simple way to specify the delegation policy is to place the same conditions on delegation as on the original role assignment. When delegation is permitted, the delegate is registered in the context store. Permissions of the role are granted to him, and tasks for the role are presented to him for the duration of the delegation.

Figure 4.28 illustrates the sequence flow of a delegation. First, the user (Bob) logs into his dashboard. The dashboard obtains the list of his current role ownerships and displays it. Bob then triggers delegation of one of his roles in a process instance to one of his colleagues, e.g., he delegates the Coach role in the process instance ”APL 0815” to Charlie. The dashboard sends the request to the PIP-IR, which forwards it to the PDP-BP. The Delegation Service retrieves the delegation policy for the Coach role from the Policy Store and determines that the delegation is allowed. The PIP-IR records Bob as the new holder of the Coach role and notifies the Re-delegation Handler (subsequent processing not shown here). The dashboard displays a confirmation to Bob. Then Bob tries to delegate his Coach role in the process instance ”APL 4711” to Dora. The sequence of messages is exactly the same until the DIS evaluates the policy and denies the delegation. This time, the dashboard displays an error message to Bob.

### 4.7.5 Constraints on user-task assignbment

Separation of duty between process roles is demanded by requirement D3.1-R.6. In the following, we explain our concept and its relation to existing approaches.

The RBAC model already specifies separation of duty (SoD) as an extension. RBAC distinguishes between static separation of duty and dynamic separation of duty. Static separation of duty is a constraint on the role assignment itself, and thus is not specific to business process management. Dynamic separation of duty restricts the activation of conflicting roles within one user session. This means that a user cannot activate conflicting roles at the same time. When interpreting ”session” as ”business-process instance”, dynamic separation of duty matches our notion of separation of duty between instance-specific roles. This is similar to the perspective of multisession separation of duties (MSoD) [91]. There, separation of duties is specified and guaranteed for business contexts or business context instances. A business context can be a single business process or a set of business processes during which an MSoD policy must persist. An MSoD policy can affect all instances of a business context or each separate instance of a business context.

In our model, constraints on user-task assignment are defined based on task groups. For the tasks within one group, binding of duty may be specified. It can be weak so that it can be overridden by re-assigning the tasks from one actor to another with the consent of both. Separation of duty is specified between groups of tasks and concerns the instantations of those tasks within one process instance.

As mentioned above, user-task-assignment rules will be attached to structured activities. Accordingly, the same will hold for constraints.

Constraints need to be enforced when users are assigned to tasks. This requires a stateful PDP-BP.

### 4.7.6 Process Execution Semantics

To accurately specify security properties of business process, a formal semantics for the business process itself is needed. For example, abstract state machines (ASMs), petri nets or operational semantics is possible.

In this section, we present our approach, an operational semantics for business processes. It takes into account the compositional structure of business process. In Section 4.7.7 below, we define security specifications based on this semantics.

Our formal model of processes focuses on the execution semantics itself, i. e., the control flow. For now, it omits external communication. However, it includes human activities as part of a process and the data involved, i. e., the organizational and informational aspect.

The model is kept simple and ignores advanced features like exception and compensation handling. As atomic processes, we support only human tasks. Other possible types include send and receive operations and data assignments with no external effect. Additionally, our model only supports nested control flow structures.[3]

**Definition 1** (Atomic and composite processes)**.** *Let $\mathcal{P}$ be the set of (possible) processes. The function $atomic : \mathcal{P} \to \mathbb{B}$ is defined for every $p \in \mathcal{P}$. $\mathbb{B}$ denotes boolean values. Atomic processes are those where $atomic(p)$ is* true*. For composite processes, $atomic(p)$ is* false*. We refer to the sets of atomic and composite processes as $\mathcal{P}_a$ and $\mathcal{P}_c$, respectively.*

**Definition 2** (Types of atomic processes)**.** *The type of an atomic process is an element of $\{ht, send, recv, nop\}$ (i. e., human task, send operation, receive operation, or empty operation). The function $type : \mathcal{P}_a \to \{ht, send, recv, nop\}$ assigns each atomic process its type.*

*We define $\mathcal{P}_h = \{p \in \mathcal{P}_a | type(p) = ht\}$ and $\mathcal{P}_{sr} = \{p \in \mathcal{P}_a | type(p) = send \vee type(p) = recv\}$.*

**Definition 3** (Types of composite processes)**.** *The type of a composite process is an element of $\{seq, alt, par, loop\}$ (i. e., sequence, alternative, parallel flow, or loop). The function $type : \mathcal{P}_c \to \{seq, alt, par, loop\}$ assigns each composite process its type.*

[3] WS-BPEL supports unstructured control-flow via control-dependencies processing, see section 11.6 of [76].

*Composite processes have one (loop) or several (all other types) member processes. For sequences, the members are ordered.*

- *For composite processes of type alt or par, a function of type members* : $\{p \in \mathcal{P}_c | type(p) \in \{alt, par\}\} \rightarrow 2^{\mathcal{P}}$ *determines the (unordered) set of members.*
- *For composite processes of type seq, a function of type size* : $\{p \in \mathcal{P}_c | type(p) = seq\} \rightarrow \mathbb{N} \setminus \{0\}$ *determines the number of members, and a function of type member* : $\mathbb{N}_0 \times \{p \in \mathcal{P}_c | type(p) = seq\} \rightarrow \mathcal{P} \cup \perp$ *determines the individual members, with* $member(i, p) = \perp$ *for* $i \geq size(p)$. *The* bottom element $\perp$ *denotes an undefined function result.*
- *For composite processes of type loop, a function of type member* : $\{p \in \mathcal{P}_c | type(p) = loop\} \rightarrow \mathcal{P}$ *determines the process that makes up the loop body.*

We need some auxiliary definitions concerning composite processes:

**Definition 4** (Containment relationship)**.** *A process* $p \in \mathcal{P}$ directly contains *another process* $q \in \mathcal{P}$ *(the predicate* $d\_contains(p, q)$ *holds) iff* $q \in members(p) \vee \exists i \in \mathbb{N} : member(i, p) = q \vee member(p) = q$.

**Definition 5** (Contained processes)**.** *The set of processes contained in a process* $p \in \mathcal{P}$ *is defined as the closure of the d_contains predicate.*

**Definition 6** (Well-formed composite processes)**.** *A process* $p \in \mathcal{P}$ *is well-formed if the directed graph formed by* $contained(p)$ *as vertices and* $\{(q, r) | q, r \in contained(p) \wedge contains(q, r)\}$ *as edges is a tree with root* $p$.

Note that, conceptually, we could relax the requirement that the graph mentioned in Definition 6 forms a tree. Instead, it would be sufficient to demand it to be a directed acyclic graph. However, this makes several definitions more complicated, as the path from the root process $p$ to an arbitrary process $p'$ becomes ambiguous. We plan to address this problem in the next version of this document.

Now that we have defined process models and their structure, we can move on to the execution semantics. Until now, we have defined process *models*. In order to execute a process an *instance* of a process model is created. There can be more than one active instance of a process model at any given time.

**Definition 7** (Process instances)**.** *Let* $\mathcal{I}$ *be the set of process instances. A function of type model* : $\mathcal{I} \rightarrow \mathcal{P}$ *determines the process model for a process instance.*

**Definition 8** (Execution state)**.** *The execution state of a process instance* $i_p \mathcal{I}$ *is a function of type* $s_e$ : $contained(model(i_p)) \rightarrow \{none, before, in, after\}$.

The values before, in, and after indicate that a process is about to be executed, is currently executing or has just finished execution. Neither of these cases holds when $s_e(p') =$ none. A process can only be active (i.e., have a state other than none) when its parent process is in state in. Composite processes pass control to either one or several of their members. Then they wait for the invoked member(s) or the sequence of members to finish.

**Definition 9** (State transition)**.** *Given a process instance* $i_p \in \mathcal{I}$*, a state transition is a pair* $(s_e^{old}, s_e^{new})$ *of execution states, where* $s_e^{new}$ *results from an atomic change to* $s_e^{old}$. *By* atomic change *we mean that only one atomic process in* $contained(p)$ *passes control to member processes or resumes control from them, or a process passes control to its immediate successor in a sequence.*

In Definition 10, we enumerate the state transitions allowed in a formal way. State transitions always apply to an atomic process $p_a \in$ contained(model($i_p$)) or to a composite process $p_c \in$ contained(model($i_p$)) and its (immediate) members. The state transitions that are possible with an atomic process are from before to in and from in to after. When a composite process $p_c$ changes its state from before to in, one or all of its members (depending on the type of $p_c$) change their state from none to before. $p_c$ can only change from in to after when all its member processes are in state none or after. For a process of type loop, it is also possible to remain in state in while its member process loops, i.e., is reset from after to before.

**Definition 10** (State transitions allowed)**.** *Let $p = model(i_p)$. We define the state transitions by expressing the new state $s'_e$ in terms of the old one, $s_e$. $s'_e(p') = s_e(p')$ for all $p' \in contained(p)$, unless we define the changed state explicitly. Only the following state transitions are allowed:*

- *Let $q \in contained(p)$, $q \in \mathcal{P}_a$ and $s_e(q) \neq none$.*
  - *$s'_e(q) = in$ if $s_e(q) = before$*
  - *$s'_e(q) = after$ if $s_e(q) = in$*
- *Let $q \in contained(p)$, $q \in \mathcal{P}_c$, $type(q) = seq$, $i \in \mathbb{N}$.*
  - *If $s_e(q) = before$: $s'_e(q) = in$, $s'_e(member(0, q)) = before$.*
  - *If $s_e(q) = in$, $i < size(q) - 1$, $s_e(member(i, q)) = after$: $s'_e(member(i, q)) = none$, $s'_e(member(i + 1, q)) = before$*
  - *If $s_e(q) = in$, $i = size(q) - 1$, $s_e(member(i, q)) = after$: $s'_e(member(i, q)) = none$, $s'_e(q) = after$.*
- *Let $q \in contained(p)$, $q \in \mathcal{P}_c$, $type(q) = alt$, $m \in members(q)$.*
  - *If $s_e(q) = before$: $s'_e(q) = in$, $s'_e(m) = before$, $\forall m' \in members(q) \setminus \{m\} : s'_e(m') = none$.*
  - *If $s_e(q) = in$, $s_e(m) = after$: $s'_e(q) = after$, $s'_e(m) = none$.*
- *Let $q \in contained(p)$, $q \in \mathcal{P}_c$, $type(q) = par$.*
  - *If $s_e(q) = before$: $s'_e(q) = in$, $\forall m \in members(q) : s'_e(m) = before$.*
  - *If $s_e(q) = in$, $\forall m \in members(q) : s_e(m) = after$: $s'_e(q) = after$, $\forall m \in members(q) : s'_e(m) = none$.*
- *Let $q \in contained(p)$, $q \in \mathcal{P}_c$, $type(q) = loop$.*
  - *If $s_e(q) = before$: $s'_e(q) = in$, $s'_e(member(q)) = before$.*
  - *If $s_e(q) = in$, $s_e(member(q)) = after$, two transitions are possible:*
    - *$s'_e(q) = in$, $s'_e(member(q)) = before$*
    - *$s'_e(q) = after$, $s'_e(member(q)) = none$*

In most cases, only one state transition is possible. The only exception is when a loop body has finished execution: The loop can continue for another iteration, or it can exit.

**Definition 11** (Execution)**.** *The execution of a process instance $i_p \in \mathcal{I}$ is a sequence $(s^0_e, s^1_e, \ldots)$ of execution states of $i_p$, where for each $s^i_e$ in the sequence with $i > 0$, the transition from $s^{i-1}_e$ to $s^i_e$ is allowed according to Definition 10.*

*Let $p = model(i_p)$.For the initial state $s^0_e$, $s^0_e(p) = before$ and $\forall q \in contained(p) \setminus \{p\} : s^0_e(q) = none$ hold. If the execution sequence is finite, the following must hold for the final state $s^n_e$: $s^n_e(p) = after$, and $s^n_e(p') = none$ for all other $p' \in contained(p)$.*

## 4.7.7 Formalisation of Security Concepts

### 4.7.7.1 Process roles

We now turn to the formal representation of process roles. To define role-assignment policies, we need some definitions concerning individuals, services and attributes.

We provide the possibility to define roles for any process. The process forms the scope in which the role is available. The level on which a role is defined is semantically significant, as it defines the duration of the role assignment and of the responsibility connected with the role.

**Definition 12** (Process roles)**.** *Let $\mathcal{R}_h$ and $\mathcal{R}_s$ be the set of possible role names for humans and (web) services, respectively. For convenience, we define $\mathcal{R} = \mathcal{R}_h \cup \mathcal{R}_s$. Functions of types $roles_h : \mathcal{P} \to 2^{\mathcal{R}_h}$ and $roles_s : \mathcal{P} \to 2^{\mathcal{R}_s}$ define the roles for humans and services used in each process.*

*We also define functions that collect all the roles defined in a composite process and its children: $roles_h^*(p) = \cup_{p' \in contained(p)} roles_h(p')$ and $roles_s^*(p) = \cup_{p' \in contained(p)} roles_s(p')$.*

**Example 4.7.7.1 Roles in the APL process**
Coach and assessor are roles in the APL process. Let $p_{\text{APL}} \in \mathcal{P}_c$. Then $\text{roles}_h(p_{\text{APL}}) \supset \{\text{coach}, \text{assessor}\}$. Furthermore, suppose that we have a sub-process for language-assessment $p_{LANG} \in \text{contained}(p_{APL})$, with $\text{roles}_h(p_{\text{LANG}}) = \{\text{lang\_assessor}\}$. Thus, $\text{roles}_h^*(p_{APL}) \supset \{\text{coach}, \text{assessor}, \text{lang\_assessor}\}$.

**Definition 13** (Association of activities with process roles)**.** *Each human task and each activity performed by a service belongs to a role. A function ht-role of type $\mathcal{P}_h \to \mathcal{R}_h$ and a function sr-role of type $\mathcal{P}_{sr} \to \mathcal{R}_s$ define this association.*

Definition 13 demands that *every* activity belongs to a role. This is viable even if the assignment is known in advance. In such cases, we can provide an initialisation for the assignment of actors to roles.

When a role is associated with an activity, it must be defined for an enclosing scope. We formalize this property in the following definition:

**Definition 14** (Well-formed task-role assignment)**.** *Let $p \in \mathcal{P}$. The task-role assignment in $p$ is well-formed iff $\forall p_h \in contained(p) \cap \mathcal{P}_h : \exists p' \in contained(p) : p_h \in contained(p') \wedge role_h(p_h) \in roles_h(p')$ and $\forall p_s \in contained(p) \cap \mathcal{P}_s : \exists p' \in contained(p) : p_s \in contained(p') \wedge role_s(p_s) \in roles_s(p')$ hold.*

**Example 4.7.7.2 Activities and roles in the APL process**
The Kenteq APL process contains a number of distinct activities performed by humans:
$\{ConductAssessment, CandidateReview, QCReview\} \subset \mathcal{P}_h \cap \text{contained}(p_{APL})$

For each of them, the function ht-role determines the associated role:

- ht-role$(ConductAssessment) = assessor$
- ht-role$(CandidateReview) = candidate$
- ht-role$(QCReview) = quality_controller$

The previous definitions have addressed roles in connection with process models, i. e. static aspects of role management. In the following, we turn to the dynamic aspects, including the assignment of actors to roles at runtime, the definition and evaluation of policies to control this assignment, and attributes needed for the evaluation of policies.

**Example 4.7.7.3 Attributes in the APL process**
Each instance of the APL process has some unique characteristics. For example, John works at MegaTools, Inc., as metalworker. Attributes of the process instance include *Industry = "Mechanical Engineering"*, *JobClassification = "Worker"*, *ClientCompany = "MegaTools, Inc."*.

**Definition 15** (Attributes)**.** *Let $\mathcal{A}$ be the set of possible attribute names and $\mathcal{V}$ the set of possible attribute values.*

*Let $\mathcal{H}$ be the set of humans. The attributes assigned to individuals are a function of type $atts_h : \mathcal{H} \times \mathcal{A} \times \mathbb{N}_0 \to \mathcal{V} \cup \{\perp\}$, and $atts_h(h, a, t)$ is the value of attribute $a$ assigned to $h$ at time $t$.*

*A similar definition holds for services: Let $\mathcal{S}$ the set of available services. The attributes assigned to services at time $t$ are a function of type $atts_s : \mathcal{S} \times \mathcal{A} \times \mathbb{N}_0 \to \mathcal{V} \cup \{\perp\}$.*

*Finally, a function of type $atts_p : \mathcal{I} \times \mathcal{A} \times \mathbb{N}_0 \to \mathcal{V} \cup \{\perp\}$ represents the attributes assigned to process instances at a certain time.*

**Definition 16** (Actor-role assignment at runtime)**.** *The actor-role assignment for humans is a function of type $h\text{-}actors : \mathcal{I} \times \mathcal{R}_h \times \mathbb{N}_0 \to \mathcal{H} \cup \{\perp\}$, and the actor-role assignment for services is a function of type $s\text{-}actors : \mathcal{I} \times \mathcal{R}_s \times \mathbb{N}_0 \to \mathcal{S} \cup \{\perp\}$.*

*This function is only defined for roles belonging to the model: h-actors*$(i_p, r, t) = \perp$ *if* $r \notin$ *roles*$^*_h(model(i_p))$, *and s-actors*$(i_p, r, t) = \perp$ *if* $r \notin$ *roles*$^*_s(model(i_p))$.

*As a shortcut, we define actors*$_p(r, t) =$ *h-actors*$_p$ *if* $r \in \mathcal{R}_h$ *and actors*$_p(r, t) =$ *s-actors*$_p$ *if* $r \in \mathcal{R}_s$.

**Definition 17** (Role assignment policies)**.** *The set of possible role-assignment policies for the role* $r \in \mathcal{R}$ *in process model* $p \in \mathcal{P}$, $\mathcal{RAP}(p, r)$, *is defined inductively. In the following, we give the rules and the meaning of the resulting policies.*

- $\forall a_p, a_a \in \mathcal{A}$ : *attequal*$(a_p, a_a) \in \mathcal{RAP}(p, r)$*: The values of the process-instance attribute* $a_p$ *and of the attribute* $a_a$ *of the prospective assignee of the role are equal.*
- $\forall a_a \in \mathcal{A}, v \in \mathcal{V}$ : *valequal*$(a_a, v) \in \mathcal{RAP}(p, r)$*: The attribute* $a_a$ *of the prospective assignee is equal to a given value.*
- $\forall pol_1, pol_2 \in \mathcal{RAP}(p, r) : (pol_1 \curlywedge pol_2) \in \mathcal{RAP}(p, r)$*: Both* $pol_1$ *and* $pol_2$ *hold.*
- $\forall pol_1, pol_2 \in \mathcal{RAP}(p, r) : (pol_1 \curlyvee pol_2) \in \mathcal{RAP}(p, r)$*: At least one of* $pol_1$ *or* $pol_2$ *holds.*

*The set of all possible role assignment policies is defined as:*

$$\mathcal{RAP} = \bigcup_{r \in \mathcal{R}, p \in \mathcal{P}} \mathcal{RAP}(p, r)$$

*Given a process* $p \in \mathcal{P}$*, a function of type* $pol_p$ : *roles*$_h(p) \cup$ *roles*$_s(p) \rightarrow \mathcal{RAP}$. *determines the assignment policies for the roles of that process.*

**Example 4.7.7.4 Role assignment policy for the coach role**
In the APL process, only coaches with expertise in the industry the candidate is working in may assume the coach role:
attequal$(Industry, ExpertiseInIndustry) \curlywedge$ valequal$(Coach, \text{“}true\text{”})$

Given a role, an actor, and the context, the function allowed determines whether the actor fulfills the applicable policy. It uses poleval to actually evaluate this policy.

**Definition 18** (Evaluation of role-assignment activities)**.** *The function poleval* : $\mathcal{RAP} \times (\mathcal{H} \cup \mathcal{S}) \times \mathcal{I} \times \mathbb{N}_0 \times (\mathcal{V} \cup \perp)^{(\mathcal{H} \cup \mathcal{S}) \times \mathcal{A} \times \mathbb{N}_0} \times (\mathcal{V} \cup \perp)^{\mathcal{I} \times \mathcal{A} \times \mathbb{N}_0} \rightarrow \mathbb{B}$*, is defined as follows:*

- *poleval*$(attequal(a_p, a_a), a, i_p, t, atts_x, atts_p) \mapsto (atts_p(i_p, a_p, t) = atts_x(a, a_a, t)) \wedge atts_x(a, a_a, t) \neq \perp$ *(with* $a_p, a_a \in \mathcal{A}$*)*
- *poleval*$(valequal(a_a, v), a, i_p, t, atts_x, andatts_p) \mapsto atts_x(a, a_a, t) = v$
  *(with* $a_a \in \mathcal{A}, v \in \mathcal{V}$*)*
- *poleval*$(pol_1 \curlywedge pol_2, a, i_p, t, atts_x, andatts_p) \mapsto$ *poleval*$(pol_1, a, i_p, t, atts_x, atts_p) \wedge$
  *poleval*$(pol_2, a, i_p, t, atts_x, andatts_p)$
  *(with* $pol_1, pol_2 \in \mathcal{RAP}$*)*
- *poleval*$(pol_1 \curlyvee pol_2, a, i_p, t, atts_x, atts_p) \mapsto$ *poleval*$(pol_1, a, i_p, t, atts_x, atts_p) \vee$
  *poleval*$(pol_2, a, i_p, t, atts_x, atts_p)$
  *(with* $pol_1, pol_2 \in \mathcal{RAP}$*)*

**Definition 19** (Authorisation of role-assignment activities)**.** *The function allowed* : $\mathcal{I} \times (\mathcal{R}_h \cup \mathcal{R}_s) \times (\mathcal{H} \cup \mathcal{S}) \times \mathbb{N}_0 \times (\mathcal{V} \cup \perp)^{\mathcal{H} \times \mathcal{A} \times \mathbb{N}_0} \times (\mathcal{V} \cup \perp)^{\mathcal{S} \times \mathcal{A} \times \mathbb{N}_0} \times (\mathcal{V} \cup \perp)^{\mathcal{I} \times \mathcal{A} \times \mathbb{N}_0} \rightarrow \mathbb{B}$ *is defined as follows*[4]*:*

*allowed*$(i_p, r, a, t, atts_h, atts_s, atts_p)$ *equals poleval*$(pol_p(r), a, i_p, t, atts_h, atts_p)$ *for* $a \in \mathcal{H}$ *and poleval*$(pol_p(r), a, i_p, t, atts_s, atts_p)$ *for* $a \in \mathcal{S}$.

[4] $A^B$ denotes the set of functions from $B$ to $A$

#### 4.7.7.2 Permission management

We can now turn to the formalization of the permission management system introduced in Section 4.7.3.

**Definition 20** (Permissions)**.** *A permission is a class of operations that can be performed on a resource.* $\mathfrak{P}$ *is the set of permissions with a partial order* $\leq$*, the implication relationship. If* $\mathfrak{p}_1 \leq \mathfrak{p}_2$*,* $\mathfrak{p}_2$ *implies* $\mathfrak{p}_1$*.*

For each process model, we define a number of resource slots. A running process instance can assign a resource to each slot at runtime. Permission allocation rules take the current resource allocation and actor-role assignment to determine the permissions to be granted to users. Resource slots define which permissions the process must possess for the resource allocated to the slot.

**Definition 21** (Resource slots)**.** *The set* $\mathfrak{S}$ *gives the available names for resource slots. A resource slot is a pair consisting of a slot name and a set of necessary permissions. A function of type* $slots : \mathcal{P} \rightarrow 2^{\mathfrak{S} \times 2^{\mathfrak{P}}}$ *specifies the resource slots for a process model.*

*For a given process* $p \in \mathcal{P}$*, we also define a function that collects all slots from child processes:* $slots^*(p) = \cup_{p' \in contained(p)} slots(p')$*.*

**Definition 22** (Resource allocation)**.** *Let* $\mathfrak{R}$ *be the set of existing resources. A resource allocation is a function of type* $alloc_p : slots^*(p) \times \mathbb{N}_0 \rightarrow \mathfrak{R}$*. It maps from slots to resources at a given time.*

Permission-allocation rules specify the permissions of actors in the process regarding the resources assigned to the process. Such rules consist of a slot referred to by its name, a role and a permission to be assigned.

**Definition 23** (Permission-allocation rules)**.** *A function of type* $rules : \mathcal{P} \rightarrow 2^{\mathfrak{S} \times \mathcal{R} \times \mathfrak{P}}$ *specifies the permission-allocation roles for all processes.*

Processes are hierarchically nested, and roles and resource slots are valid in all sub-processes contained in a process. We must ensure that role names and resource slot names uniquely identify roles and resource slots, respectively. In order to achieve this, two alternatives are available: Either, roles and resource slots defined in a process hide those defined in a higher-level process, or re-using names already used in a higher-level process is not allowed. To keep things simple, we choose the latter alternative.

Permission-allocation rules reference definitions of roles and resource slots. In order for such references to be meaningful, the definition must occur in the same process it is used in, or a higher-level one.

**Definition 24** (Consisteny criteria for processes)**.** *A process* $p \in \mathcal{P}$ *fulfills the following consistency criteria:*

- *No hiding of names used in a higher-level process:*
  - $\forall r \in \mathcal{R}_h : \forall p' \in contained(p) \setminus \{p\} : r \in roles_h(p) \rightarrow r \notin roles_h(p')$
  - $\forall r \in \mathcal{R}_s : \forall p' \in contained(p) \setminus \{p\} : r \in roles_s(p) \rightarrow r \notin roles_s(p')$
  - $\forall s \in \mathfrak{S} : \forall p' \in contained(p) \setminus \{p\} : (\exists \mathfrak{p} \in \mathfrak{P} : (s, \mathfrak{p}) \in slots(p)) \rightarrow (\neg \exists \mathfrak{p}' \in \mathfrak{P} : (s, \mathfrak{p}') \in slots(p'))$
- *Definition of used roles and resource slots:*
  - $\forall p' \in contained(p), s \in \mathfrak{S}, r \in \mathcal{R}, \mathfrak{p} \in \mathfrak{P} : (s, r, \mathfrak{p}) \in rules \rightarrow (\exists p'' \in contained(p), \mathfrak{p}' \in \mathfrak{P} : p' \in contained(p'') \wedge (s, \mathfrak{p}') \in slots(p'') \wedge (\mathfrak{p}' \geq \mathfrak{p}))$
  - $\forall p' \in contained(p), s \in \mathfrak{S}, r \in \mathcal{R}, \mathfrak{p} \in \mathfrak{P} : (s, r, \mathfrak{p}) \in rules \rightarrow (\exists p'' \in contained(p), \mathfrak{p}' \in \mathfrak{P} : p' \in contained(p'') \wedge r \in roles(p''))$

**Definition 25** (Allocation results)**.** *Given a process instance $i_p \in \mathcal{I}$ and a point in time $t$, let h-actors and s-actors be the role assignment, $alloc_p$ the resource allocation, and $s_e^t$ the execution state of the process instance.*

*The allocation result is the set of permissions valid at $t$, namely:*

$\bigcup_{p' \in \{p'' \in contained(p) | s_e^t(p'') = in\}} \bigcup_{(s,r,\mathfrak{p}) \in rules(p')} : \{(actors(i_p, r, t), alloc_p(s, t), \mathfrak{p})\}$.

Definition 25 collects permission-allocation rules from all *active* sub-processes of a given process (i.e., sub-processes with an execution state of in). The result consists of the instantiation of all rules, i.e., resource slots and roles are filled in with actual resources and actors taken from the assignment valid at $t$.

#### 4.7.7.3 Extended Execution Semantics

We have introduced an execution semantics at the beginning of this section. In the following, we present an extended execution semantics that includes the current security context, i.e., the assignment of roles, attributes and resources to process instances.

We need explicit actions to trigger these assignments. Thus, we need to introduce new types of atomic operations and adapt Definition 2 accordingly.

**Definition 26** (Types of atomic processes (revised))**.** *The type of an atomic process is defined by the function type $: \mathcal{P}_a \to \{ht, send, recv, ar, sa, ar, nop\}$. Atomic processes with the new types aa, sa, and ar assign actors (humans or services) to roles, set attribute values of the process instance, and allocate resources, respectively.*

*A function of type aa-role $: \{p \in \mathcal{P}_a | type(p) = aa\} \to (\mathcal{R}_h \cup \mathcal{R}_s)$ returns the role which the operation assigns actors to. A function of type aa-actor $: \mathcal{I} \times \mathbb{N}_0 \to (\mathcal{H} \cup \mathcal{S}) \cup \bot$ determines the actor assigned by the operation at a given time when the process is actually executed.*

*sa-att $: \{p \in \mathcal{P}_a | type(p) = sa\} \to \mathcal{A}$ is the type of a function that returns the attribute set by the operation. A function of type sa-val $: \mathcal{I} \times \mathbb{N}_0 \to \mathcal{V} \cup \bot$ gives the value assigned to the attribute at time $t$.*

*Finally, a function of type ar-slot $: \{p \in \mathcal{P}_a | type(p) = sa\} \to \mathfrak{S}$ specifies which slot an operation of type ar assigns resources to, while a function of type ar-res $: \mathcal{I} \times \mathbb{N}_0 \to \mathfrak{R} \cup \bot$ specifies the actual resource assigned at time $t$.*

**Example 4.7.7.5 Assignment of an actor in the APL process**

In the first phase of the APL process, the process chooses a coach and an assessor. We look at a simplified process $p$ consisting only of the relevant activities.

$p$ is a sequence, $p \in \mathcal{P}_c$ and type$(p)$ = seq. It consists of two sub-processes that assign users to roles: size$(p) = 2$, member$(0, p) = p_0$, member$(1, p) = p_1$, $p_0, p_1 \in \mathcal{P}_a$, type$(p_0)$ = type$(p_1)$ = ar.

$p$ defines two roles, coach and assessor: roles$_h(p) = \{$coach, assessor$\}$. $p_0$ and $p_1$ assign actors to the coach and to the assessor role, respectively: aa-role$(p_0)$ = coach, aa-role$(p_1)$ = assessor.

Let $i_p$ be an instance of $p$, model$(i_p) = p$. Let execution of $p_0$ and $p_1$ in $i_p$ occur at $t_0$ and $t_1$, respectively. If aa-actor$(i_p, t_0)$ = Bob and aa-actor$(i_p, t_1)$ = Jim, Bob will be assigned to the coach role and Jim will be assigned to the assessor role after the execution of $p$.

We can now extend our execution semantics (Definition 11) so that it not only encompasses the "bare" execution state $s_e^t$ of a process instance $p$, but also the current assignments of actors to roles, resources to slots, and the current values of attributes expressed by functions h-actors$_p$, s-actors$_p$, alloc$_p$, and atts$_p$.

As already defined in Definition 11, the execution of a process $p \in \mathcal{P}$ is a sequence $(s_e^0, s_e^1, \ldots)$ of execution states of $p$. The initial state $s_e^0$ has already been defined there.

The security configuration also encompasses the assignment of actors to roles, of resources to resource slots, and the values of process instance attributes. In the following definition, we will define the initial security configuration.

**Definition 27** (Initial security configuration)**.** *Initially, all roles are unassigned, i. e., h-actors$(i_p, r, 0) = \bot$ for all $r \in roles_h^*(model(i_p))$ and s-actors$(i_p, r, 0) = \bot$ for all $r \in roles_s^*(model(i_p))$. Further, all slots are unassigned initially, i. e., $alloc_p^*(s, 0) = \bot$ for all $s \in slots^*(p)$. Finally, attributes of the process are initially unassigned as well, atts$(i_p, a, 0) = \bot$ for all $a \in \mathcal{A}$.*

Now we can extend Definition 10 with the changes of the security configuration that are allowed.

**Definition 28** (Allowed changes of the security configuration: Role assignment)**.** *Role assignments only change when an operation (atomic process) of type aa is executed, i. e., changes its state from before to in*[5]*. Furthermore, the role assignment only changes when the role-assignment policy is fulfilled. Otherwise, an error situation occurs which the process must handle.*

*Unless otherwise defined, h-actors*$(i_p, r, t)$ = *h-actors*$(i_p, r, t-1)$ *for all* $r \in$ *roles*$^*_h(model(i_p))$ *and s-actors*$(i_p, r, t)$ = *s-actors*$(i_p, r, t-1)$ *for all* $r \in$ *roles*$^*_s(model(i_p))$.

*Let* $p' \in$ *contained*$(model(i_p))$ *with type*$(p')$ = *aa,* $s_e^{t-1}(p')$ = *before,* $s_e^t(p')$ = *in. Without loss of generality*[6]*, let* $r :=$ *aa-role*$(p) \in \mathcal{H}$*. If* $allowed(i_p, r,$ *aa-actor*$(p,t), t, atts_h, atts_p)$*, then h-actors*$_p(i_p, r, t)$ = *aa-actor*$(i_p, t)$.

**Definition 29** (Allowed changes of the security configuration: Resource allocation)**.** *Resource allocations only change when an operation (atomic process) of type ar is executed. Unless defined otherwise,* $alloc_p(s,t) = alloc_p(s, t-1)$ *for all* $s \in$ *slots*$^*(p)$.

*Let* $p' \in$ *contained*$(p)$ *with type*$(p')$ = *ar,* $s_e^{t-1}(p')$ = *before,* $s_e^t(p')$ = *in. Let* $s^* :=$ *ar-slot*$(p')$*. Then* $alloc_p(s^*, t) =$ *ar-res*$(i_p, t)$.

**Definition 30** (Allowed changes of the security configuration: Process-instance attributes)**.** *Finally, attributes of process instances only change when an operation (atomic process) of type sa is executed. Unless defined otherwise,* $atts(i_p, a, t) = atts(i_p, a, t-1)$ *for all* $a \in \mathcal{A}$.

*Let* $p' \in$ *contained*$(p)$ *with type*$(p')$ = *sa,* $s_e^{t-1}(p')$ = *before,* $s_e^t(p')$ = *in. Let* $a^* :=$ *sa-att*$(p')$*. Then* $atts(i_p, a^*, t) =$ *sa-val*$(i_p, t)$.

#### 4.7.7.4 Separation of Duty

**Definition 31** (Separation-of-duty rules)**.** *A separation-of-duty rule is a pair* $(r_1, r_2) \in \mathcal{R} \times \mathcal{R}$ *and specifies a conflict between two roles. For each process p, separation-of-duty rules can be defined between roles in this process or its sub-processes. The set of such roles for a process p is determined by a function of type* $sod$ : *contained*$(p) \to 2^{\mathcal{R}\times\mathcal{R}}$

*These roles are subject to the following consistency constraints that must hold for any* $(r_1, r_2) \in sod(p')$ *for all* $p' \in$ *contained*$(p)$*:*

- *Both roles must be of the same type:* $(r_1 \in \mathcal{R}_h \wedge r_2 \in \mathcal{R}_h) \vee (r_1 \in \mathcal{R}_s \wedge r_2 \in \mathcal{R}_s)$
- *Both roles must be defined in* $p'$ *or its sub-processes:* $r_1, r_2 \in$ *roles*$^*_h(p') \cup$ *roles*$^*_s(p')$

*We define a function* $sod*$ : *contained*$(p) \to \mathcal{R} \times \mathcal{R}$ *that collects all rules defined for a composite process:* $sod^*(p) = \cup_{p' \in contained(p)} sod(p')$.

Given the definition of separation-of-duty rules, we have to adapt the evaluation of role-assignment policies in Definition 19.

**Definition 32** (Evaluation of role-assignment policies including separation of duty)**.** *Let* $p \in \mathcal{P}$ *be a process,* $r \in \mathcal{R}_h \cup \mathcal{R}_s$ *a role,* $a \in \mathcal{H} \cup \mathcal{S}$ *the human or the service to be assigned to the role (with either* $r \in \mathcal{R}_h \wedge a \in \mathcal{H}$ *or* $r \in \mathcal{R}_s \wedge a \in \mathcal{S}$*), t the current point in time, and* $atts_h$*,* $atts_s$*, and* $atts_p$ *the attribute assignment for human actors, services, and the process instance of p, respectively.*

*The function allowed'* : $\mathcal{I} \times (\mathcal{R}_h \cup \mathcal{R}_s) \times (\mathcal{H} \cup \mathcal{S}) \times \mathbb{N}_0 \times (\mathcal{V} \cup \perp)^{\mathcal{H}\times\mathcal{A}\times\mathbb{N}_0} \times (\mathcal{V} \cup \perp)^{\mathcal{S}\times\mathcal{A}\times\mathbb{N}_0} \times (\mathcal{V} \cup \perp)^{\mathcal{I}\times\mathcal{A}\times\mathbb{N}_0} \to \mathbb{B}$ *determines whether a may be assigned to r.*

*allowed'*$(i_p, r, a, t, atts_h, atts_s, atts_p)$ = *allowed*$(i_p, r, a, t, atts_h, atts_s, atts_p) \wedge (\forall (r, r_2) \in soa^* : a \neq actors_p(r_2, t)) \wedge (\forall (r_1, r) \in soa^* : a \neq actors_p(r_1, t))$

*The definitions of the functions allowed and poleval are the ones from Definition 19.*

[5] This choice of the exact point where the execution takes effect is somewhat arbitrary. What is important, however, is to make sure that the execution takes effect *at all*.

[6] For aa-role$(p) \in \mathcal{H}$, replace h-actors$_p$ by s-actors$_p$ and atts$_h$ by atts$_s$ in the following.

In order to include the enforcement of separation of duty in the process execution, we have to replace allowed by allowed' in Definition 28.

## 4.8 Efficiency and Limitations of the Approach

In this section, we analyze the efficiency of our overall approach, consisting of a security annotation language, an automated transformation, and a runtime-enforcement architecture, and possible limitations of this approach.

### 4.8.1 Security-Annotation Language

As evaluated in D3.3, we have used the security annotation language for three demonstrators in the e-employability domain, but also in an e-health application in a Break-the-Glass scenario. The applications are using personally identifiable data and support end-users with applications on their data using external web services, i.e., typical application domains on which the project focuses. We have developed the language to support those applications and have shown that the language supports the constraints required. A user study, described in D3.3 Section 6.4, has shown, that the language is flexible enough for other application areas as well: The users modelled security constraints for scenarios we never had in mind.

If there will come up new security and trust constraints, the language may need to be enhanced. We have shown such a procedure for the case of BTG in business processes, which requires to introduce an additional security annotation element. In consequence, we have had to determine the transformation of these annotations, defining the rules of the business process policy (i.e., BTG rules) and process fragments to achieve the dynamic handling of BTG situations, e.g., the BTG action itself, or the obligations to be hold. The transformation of security annotations is very flexible as it uses three targets. The adaptation of the business-process model supports new constraints and properties by easily developing process fragments for any new annotation type. The transformation to the business policy uses standard policy languages like XACML or Permis, which facilitates this step. The third target results in configuration parameters and context information for calling the security components of the security framework. As we have defined the architecture of a secure business-process-management system enhancing a WfMC architecture of a BPMS with XACML standard components, we achieved a well-founded set of target components.

There are some limitations of the language, which will need further investigations. The auditing annotation must allow to define flexible logging policies. In the current version, we allow to use such a policy (setting a policy name), but defining an audit policy has been out of scope of our work. Further, there might come up conflicts between constraints. E.g., think of SoD constraints in the same part of the process as BoD constraints might result in a conflict. There exist approaches in literature, e.g., [92], to detect such situations at modelling time or as early as possible during execution time. It would be very helpful to include such mechanisms into our approach, as well, to detect and possibly anticipate those conflicts.

### 4.8.2 Flexibility and Implementation Challenges

We evaluated our approach with respect to the demonstrator scenarios and their implementation (see Chapter 6 of D3.3). The main advantages of our approach is that it provides generic solutions for common problems that business-process deployments in service-oriented architectures face. This also allowed us to refine the mechanisms somewhat independently of the demonstrator business processes as the evolution of the trust network progressed. On the other hand, the generic mechanisms adds complexity that does not necessarily pay off for only one demonstrator.

A strength of business processes in general, not limited to our approach of *secure* business processes, is high-level integration of loosely coupled components. As elaborated in D3.3, this was particularly useful to quickly get demonstrators running, and change the order of steps afterwards. The backside of this strength can also be seen as a limitation: While business-process modelling also *supports* lower-level integration (i.e., detailed interfaces and specifying the data flow between them on a quite technical level), this does not scale well.

Our approach separates application-specific functionality and generic security functionality. For example, we have identified a set of business-process fragments for security-specific user interactions such

as service selection and accepting terms and conditions. This reduces the complexity of application processes, by simply expressing required security functionality through an annotation. Of course, the effort needed to identify process fragments that are typically needed, and to implement them, does not pay off for a few demonstrators. While we are confident that we have described a representative set of security-specific user interactions, further refinement with more, and more complex, use cases is needed.

In several cases, we have designed dedicated components implementing security functionality. In some cases, these components implement functionality a business process in a conventioal BPMS could not provide. Examples are cryptographic encryption and signing of web-service calls or single sign-on (SSO) to the user interfaces. The functionality provided is of course limited to what is actually implemented, but clearly an advantage beyond the state of the art. In other cases, we provide components for tasks that a business process could somehow perform itself in a conventional BPMS. For example, we enforce history-based constraints by recording the execution history of process instances and taking decisions based on declarative policies. Flexibility is limited by the features of the policy language used, whereas in the business process itself, arbitrary computations are possible. However, this is actually desirable for security specifications, because allowing overly complex specifications creates the risk of mistakes going unnoticed.

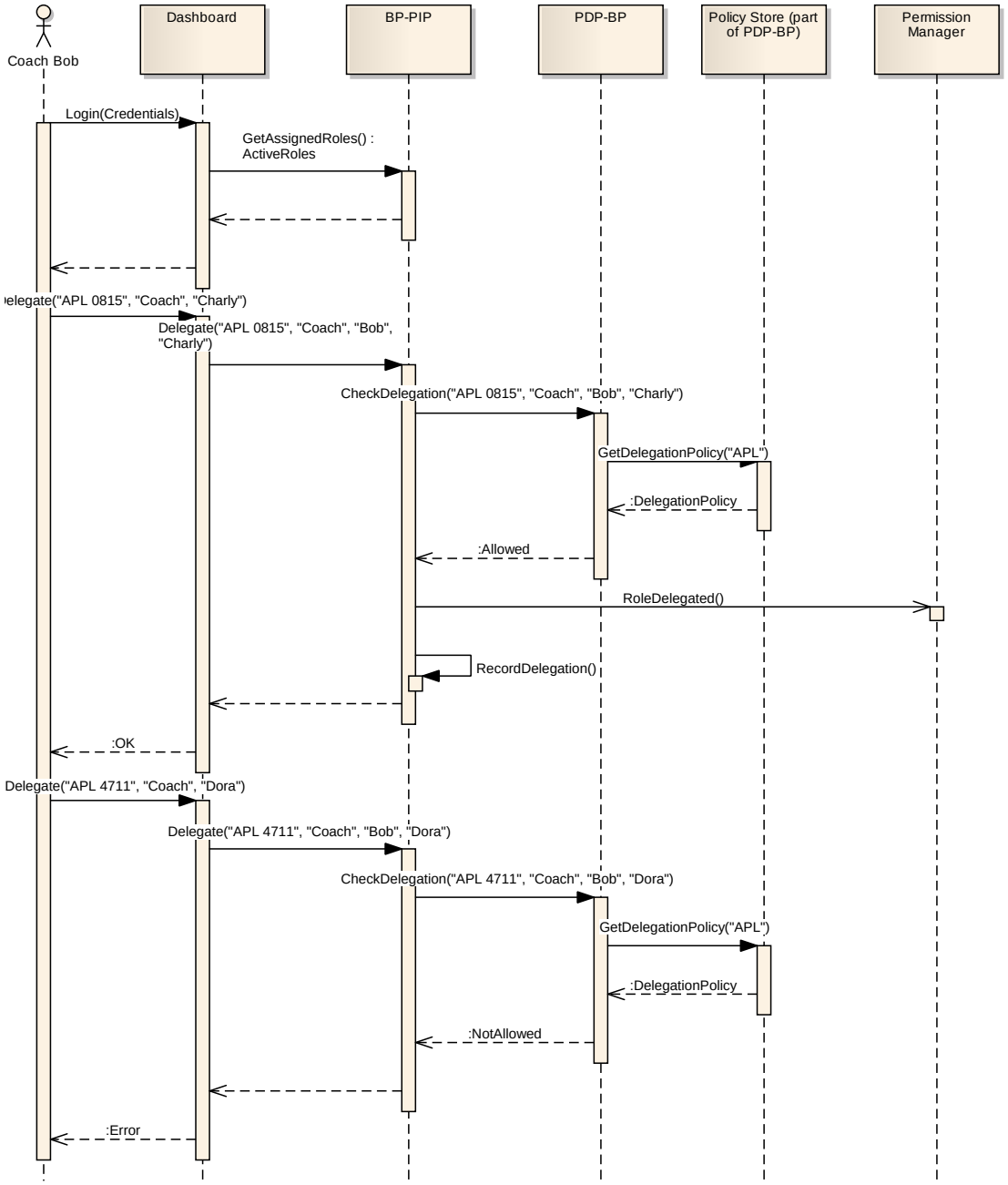


**Figure 4.28: UML sequence diagram showing the delegation of assigned tasks in a business process instance**

# 5 Implementation Design

Our security concepts are designed to integrate with existing technology (BPMN, BPEL, BPEL4People) and software (the Intalio open-source components) in the business process management domain, and with the TAS$^3$ security architecture. In this chapter, we provide the implementation design of first parts of the conceptual design. The concepts of chapter 4 are mapped to concrete implementation and we already started to prepare the validation of the concepts demonstrated with a business process of our pilot applications, the Kenteq APL process.

At the beginning, we describe the integration approach of security management for business processes in more detail. Briefly the approach to federated identity and single sign on for users of business processes is introduced. Then, a main part examines implementation of role handling of process roles with instance-specific assignments. Further parts deal with delegation and separation of duty, and fault handling mechanisms for reacting to security violations.

Finally, there are two topics. One is a very first description on security modelling during business process design and how to transform it to policies at the process execution level. Processes consist of subprocesses, which reflect a processing unit from the business point of view. With that we can handle subprocesses as scopes of security. The last topic describes the efforts on adapting processes, which are active. This results in a basic formalized model of process changes, on the schema level and on adapting process models. The adaptation architecture includes a repository of processes and process patterns. Additionally, we have started to implement an authorization module for processes and process adaptations, which was designed in previous work [93] in our group. With that we have been able to start with the main research in adaptability, namely to investigate how to use process-specific security using process adaptation. The implementation design described in this report gives an overview. For further details and further development we refer to the Deliverable D3.2 [2], the description and documentation of the software implementing the components of a secure business-process-management system.

## 5.1 Integration of Security Management for Business Processes into TAS$^3$

In this section, we describe how we will coordinate the development of the secure process management platform with the development of the TAS$^3$ infrastructure and the applications processes running on the platform.

Business processes model both (payload) applications and secure transactions in the TAS$^3$ architecture. The TAS$^3$ architecture orchestrates the security components by a well-defined, finite state diagram that can be modelled as a process. This can provide early prototypes of security processes (for production use, the performance of such an implementation will probably not suffice).

We can think of different levels of integration of (payload) business processes into the TAS$^3$ architecture. The basic level is for the process to speak the TAS$^3$ wire protocol on any web service calls it makes or receives, and to use the TAS$^3$ single-sign-on mechanism to authenticate user interactions. The wire protocol would be implemented by using a custom network stack which is transparent to the business process. Access control cannot take that state of the business process and its context into account, and the process does not even notice access-control decisions made on its behalf. Our integration approach is based on a combination of enhancements in the application processes and in the BPEL executor. We introduce process-specific security concepts handled by dedicated runtime components. The rationale is that we do not need to explicitly model them in application processes. Annotations on application processes provide the configuration for these components. Further, for cases where such implicit, modelling-time configuration is not possible, we provide application processes with the possibility to explicitly manipulate their security context. This holds for decisions affecting the security context that are highly application-dependent, possibly even involving manual steps, such as the assignment of actors in the Kenteq APL process.

Integration into the TAS$^3$ architecture is necessary in several fields:

- Single-signon: Components that interact directly with the user must use a single-singon mechanism compatible with the TAS$^3$ architecture to authenticate the user. This concerns the presentation of human tasks to actors in business processes.
- Identity management: Processes must use identity tokens in the format used by the trust network, and possibly use a mapping service when making outgoing calls on behalf of a user.
- Secure communication: The BPEL engine must transform all outgoing and incoming communication to/from the wire protocol used in the trust network.
- Authorization: The PEP used by the BPEL engine must collect context information needed for authorization and create a request that the PDP understands. The same holds for the user interface.
- Permission management and policy deployment: Permissions granted to the service provider running the business process are handled semi-automatically by components associated with the BPEL engine. It will become necessary to cause permissions to be granted to actors involved in a process. Further, policies may be instance-specific and must be automatically created when a process instance is created. Accordingly, proper interfaces must be in place to deploy credentials granting permissions and policies to the policy decision points.

## 5.2 Federated identity and single sign-on for the user interface

TAS$^3$ uses a common single sign-on (SSO) framework based on SAML 2.0. Users are known to an identity provider (IdP). Whenever they login to a service provider (SP), the identity provider is involved to assert the user's identity.

SSO support is necessary for all components involved in direct interaction with the user. This holds for the T3-BP-CLIENT component, which is based on Intalio Tempo: It provides a task-list console as a user interface. When a user logs into Tempo, he will see tasks he is eligible to execute. Two modules of Tempo are relevant here: The User Interface Framework (UIFW) handles the presentation of the user interface, including log-in and the selection of tasks. The Security Framework (SFW) provides authorization and authentication. The basic implementation in the open-source edition of Tempo compares credentials (username and password) with a simple XML file, and provides authorisation based on roles provided in that file.

In order to support SSO, we will modify the log-in functionality of the UIFW component: Instead of generating a log-in screen itself, it will forward the browser of the user to the ZXID service provider servlet (provided by Symlabs), which allows the user to choose his identity provider and log-in using that IdP. The ZXID servlet then stores session information in the Tomcat servlet container which is then used by Tempo. As both Tempo and the ZXID SP servlet run in the same Tomcat servlet container, they can use the session store to exchange information.

The identity information acquired about business process participants by the user interface is kept within the T3-BP-PEP component. For web-service calls, the PEP determines based on the business-process policy which identity to use for the call (see also Section 5.7) and provides the respective tokens.

## 5.3 User-task assignment and constraints

In this section, we will take a look at how the concept of process roles introduced in Chapter 4 strengthens the security of business processes, and in particular Kenteq's APL process, and how it fits this use case. Then, we will consider implementation issues concerning the Intalio system, including ideas about how to use the existing modelling tool to support the new security concepts. Finally, we briefly introduce possible future enhancements, namely local roles and automatic assignment of users to roles.

With the process role approach, we are able to support constraints on the activation of a role for a business process instance that take the characteristics of that instance into account. This was not possible before: For example, when a Kenteq employee logs into the system and activates his Coach role, there is

no guarantee that he actually has the knowledge to provide advice to the candidate in that instance of the APL process. Further, we can support different application needs as to when and how users are assigned to roles in the application process: Some assignments are already fixed when the process starts. For example, the candidate is in the centre of Kenteq's APL process. She is, of course, determined in the contract and is known when the APL process starts. The organiser, who is mainly charged with administrative duties, is also fixed in the contract. In other cases, the process itself (by an explicit activity) or a user performs the assignment when the process is already running. E.g., after the APL process has started, the organiser assigns a coach, an assessor, and a quality controller for the specific instance. Our proposed concept of process roles implies binding of duty. This means that the specific person holding a role in the process instance performs all tasks assigned to the role. This is also corresponds to the Kenteq APL process, and there was no explicit support for binding of duty in Intalio|BPMS before.

Currently, tasks for humans or BPMN lanes containing such tasks include the role which a user must possess in order to work on the task. This allows collecting the roles involved in a business process definition. Some roles, however, will only occur in subprocesses and not in the main process. In order to have a better overview of all roles in a process, we will explicitly include a list of all roles in the model of a business process. Each role gets the assignment policy annotated. In the process model in Figure 5.1, simple logical expressions symbolise the policies.

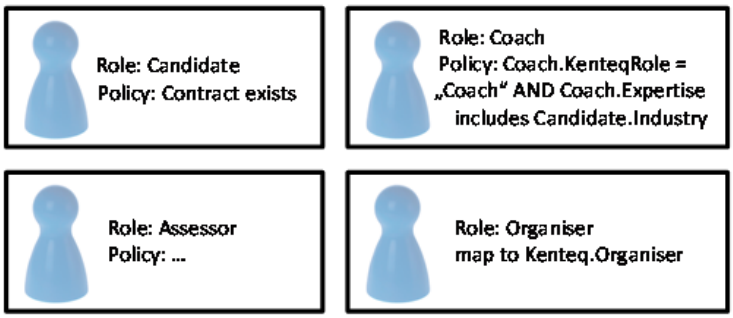


**Figure 5.1: Specification of roles and assignment policies in the APL process**

As we have already mentioned, one way of assigning users to roles in the Kenteq APL process is by explicit actions of the process, based on the decision of the human participant holding the Organiser role. This is shown in Figure 5.3, which shows the Allocate Resources scenario from the Commence APL phase: First, the Organiser is explicitly asked to name users for the different roles. Then the process calls the context store to perform the assignment.

In addition to the new components already mentioned in Chapter 4, we also have to adapt the Intalio system. This mainly concerns the task-list component (Intalio|Tempo). We will re-use the existing interfaces. When a business process creates a task for a human, it sends a request to the Intalio Tempo component. This request includes the role that has to execute the task. Currently, Tempo looks up the users having this role. Tempo has an extensible architecture and security modules that can obtain the necessary information from either a simple XML file or an LDAP server. Then it presents the task to all users found. Any of them can claim the task and carry it out. We will enhance the interface to the security module so that it contains information on the context of the task. In particular, this is the ID of the process instance it belongs to. We will introduce a new security module that involves the Business Process Policy Information Point (BP-PIP) when determining the user whom the task will be presented to. It will look up the role assignments for the process instance in question and only return the user who has been assigned to the role. This check takes place when the list of open tasks is displayed to a user.

We will provide a tool where the process designer can specify the roles occurring in the process and the corresponding assignment policies. The implementation will use XACML to encode the policies, as recommended by the TAS$^3$ architecture specification (Section 9.3.2 of Project Deliverable D2.1 [23]). For deployment to the BP-PIP, the designer has to specify the ID of the process model that the specification belongs to. In the future, we will directly integrate this view into Intalio Designer.

As process models are split up into several levels of sub-processes, we believe that these hierarchical

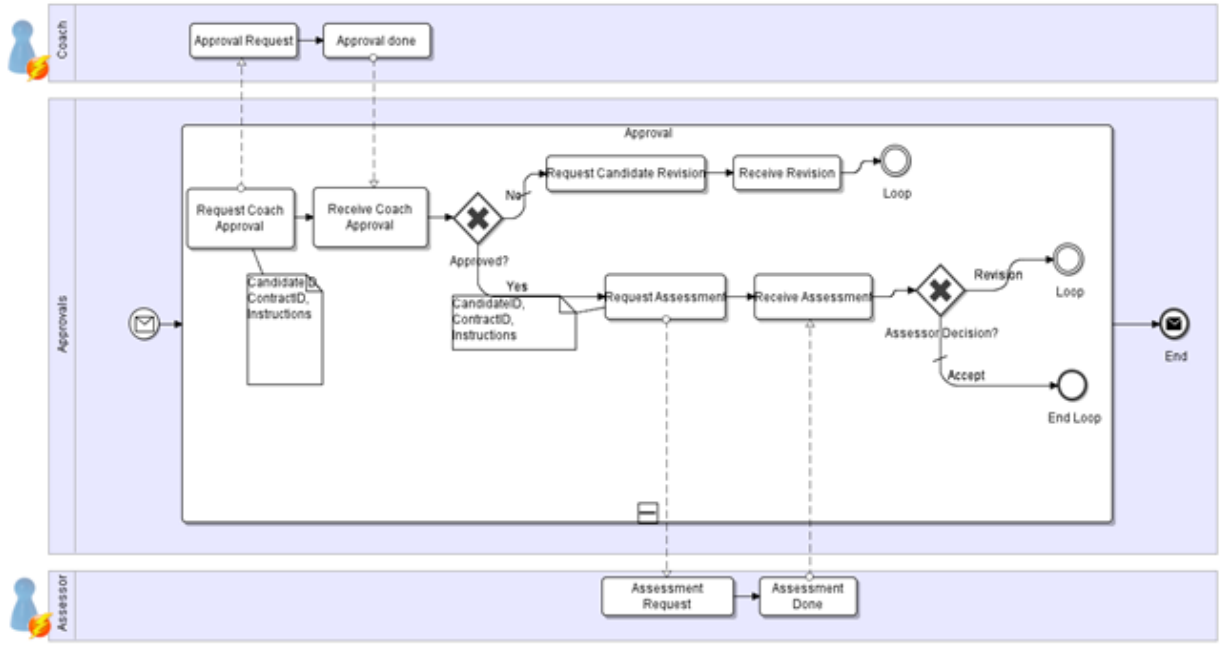


**Figure 5.2: Example use case for separation of duty**

models can be leveraged for the definition of roles as well, in line with the goal to leverage the sub-process structure for security modelling: The current version of the Kenteq APL process only contains global roles that apply to the entire process (see Figure 5.6). As an example for a possible local role, consider the case that the profile of the candidate indicates that he knows a foreign language. A sub-process to obtain a rating according to the Common European Framework of Reference for Languages is invoked for each language. This sub-process specifies a role ”language assessor”, which will be assigned to someone with knowledge of the language in question. We will consider this case in a future iteration.

The BP-PIP will provide a web-service interface to assign users to roles. The process can use this interface to perform explicit assignments. In some cases, it is also possible to let the execution engine choose an individual for a role. Different strategies are conceivable as long as the assignment policy for the role in question is fulfilled. The system can then choose either an eligible actor itself and assign the tasks to him, or it can present the first task for the role to all actors eligible and choose the first individual who claims the task. We still have to explore how to implement automatic assignment. For the time being, the process has to explicitly assign users to a role before it creates any task for the role. This means that the process designer has to include activities that first determine a user to assign to the role, and then perform the actual assignment.

The Business Process PIP component handles the assignment of actors to roles in business-process instances. It stores role definitions (originating from the modelling tool) for each business-process *model* as configuration.

Actors are identified by persistent SAML NameIDs (for humans) and endpoint references (for web-services). Discovery of suitable actors is not part of the functionality of this component. Currently, an explicit request (by the business process) is necessary to execute the assignment of an actor to a role. We are planning to extend this by a mechanism which calls a discovery component if needed.

The BP-PIP component keeps part of the security-relevant state information of business-process instances. It is not involved in the execution of business processes or in the presentation of the user interface.

The T3-BP-PIP is running as a web service and provides its functionality via a web service interface (SOAP over HTTP). It keeps its state in an embedded database.

When deploying a process model to the process engine, the BP-PIP gets the role definitions of the process model. When an instance of a business process is being created the BP-PIP receives a message from the process engine with an intended role-user assignment. It calls the PDP to check if this assignment

is permitted according to policies of the process model. If this is the case the BP-PIP stores the assignment and forwards this information to the T3-BP-PPM component.

Figure 5.4 shows the components the BP-PIP is interacting with.

During process execution, the roles used in tasks executed by the process must be translated into the assigned user or service. For human tasks, this translation happens in the process itself: The process looks up the user assigned to a role by making a request to the BP-PIP Then, it requests Tempo to create the tasks. This request already includes the actual user who has to perform the task. With the Intalio Designer modelling tool, tasks can be connected with roles. We will provide an automatic translation so that the process performs the lookup of the user assigned to the role.

For tasks involving services, the PEP-BP acts as a proxy between the business process and the actual service. The request to the PEP-BP includes the actual payload and the role of the service to be called. The PEP-BP looks up the assigned service endpoint and forwards the payload to that service.

An example of the SOAP payload of a request to the T3-PEP-BP is given in Figure 5.5. The actual payload is encapsulated inside the `<pep:payload>` element.

## 5.4 Interval Monitoring

To determine the validity of permissions for our context-aware permission management system (see Section 4.7.3), we will provide an interval monitoring component. In this section we will explore what functionality for such a component is possible in a BPEL engine, specifically Apache Ode, and give an overview of its design. The implementation will be part of the next iteration of Deliverable D3.2.

Many BPEL engines, such as Apache ODE, have an event model that allows monitoring the events related to a business process and its instances. The event model specifies a number of events that can occur during the lifecycle of a business process or its activities. Components can register and are then informed about events occuring. For example, the event handler is called when an activity is about to be executed.

The information associated with events allows to determine to which process, process instance or activity it belongs. For each instance, the type, process name, process identifier and process instance identifier is available. In addition, a timestamp and the line number in the BPEL code of the process is provided. For some event types, further information is available. For example, `ProcessInstanceStartedEvent` contains a field named `RootScopeId`. Events relating to scopes contain the name of the sorrounding scope, of the scope concerned, and the scope identifier of it. Events relating to activities contain the name and the identifier (unique within a given process instance) of the activity.

An important question is whether the execution of activities can be appropriately monitored when they are nested, executed repeatedly, or when branches of the control flow occur. In a `<sequence>`, the nested activities can be monitored exactly if their names are unique. In an `<if>` element, where either the main activity or the one contained in the `<else>` child element is executed, an activity might never be executed. Thus, intervals starting or ending in one of the alternative paths of an `<if>` are not allowed.

In looping activities, such as `<while>`, `<repeatUntil>` and `<forEach>` the status of intervals can be determined properly if the intervals do not span multiple iterations of the loop.

The event listener will share its state with the actual interval monitor web service through a common database.

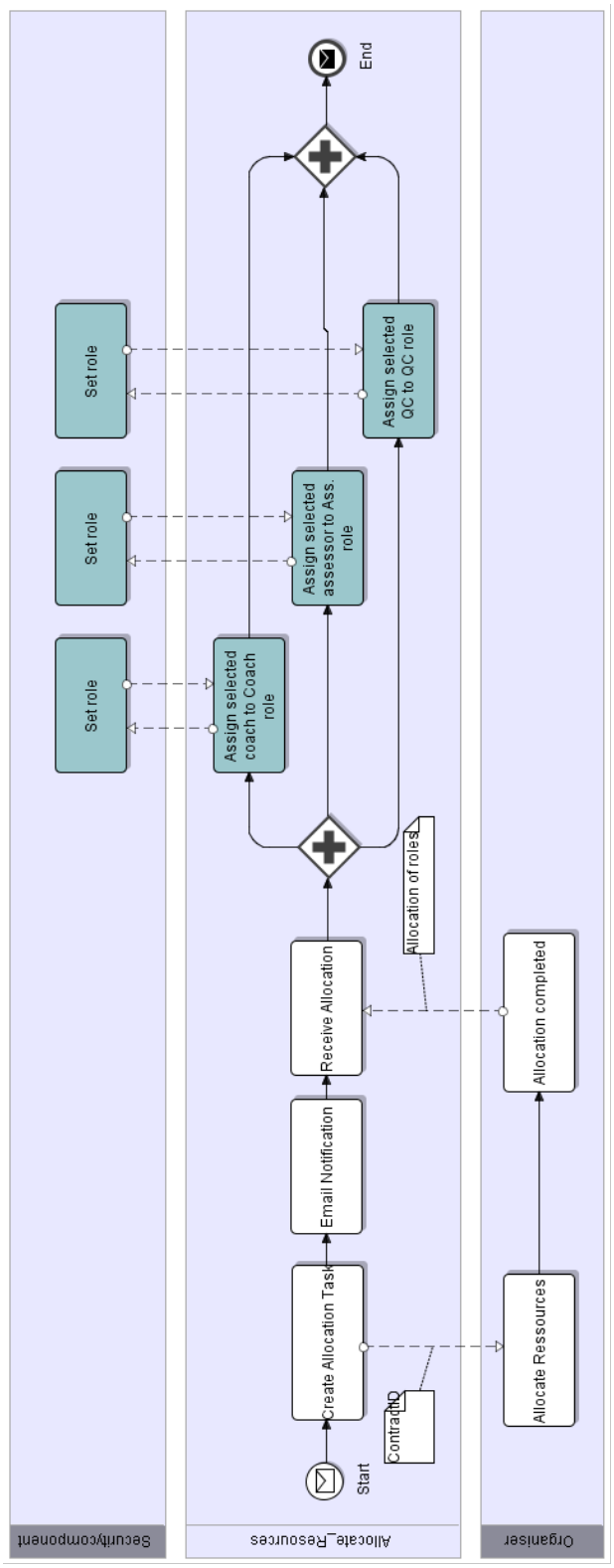


**Figure 5.3: Explicit assignment of individuals to roles based on the decision of a human actor**

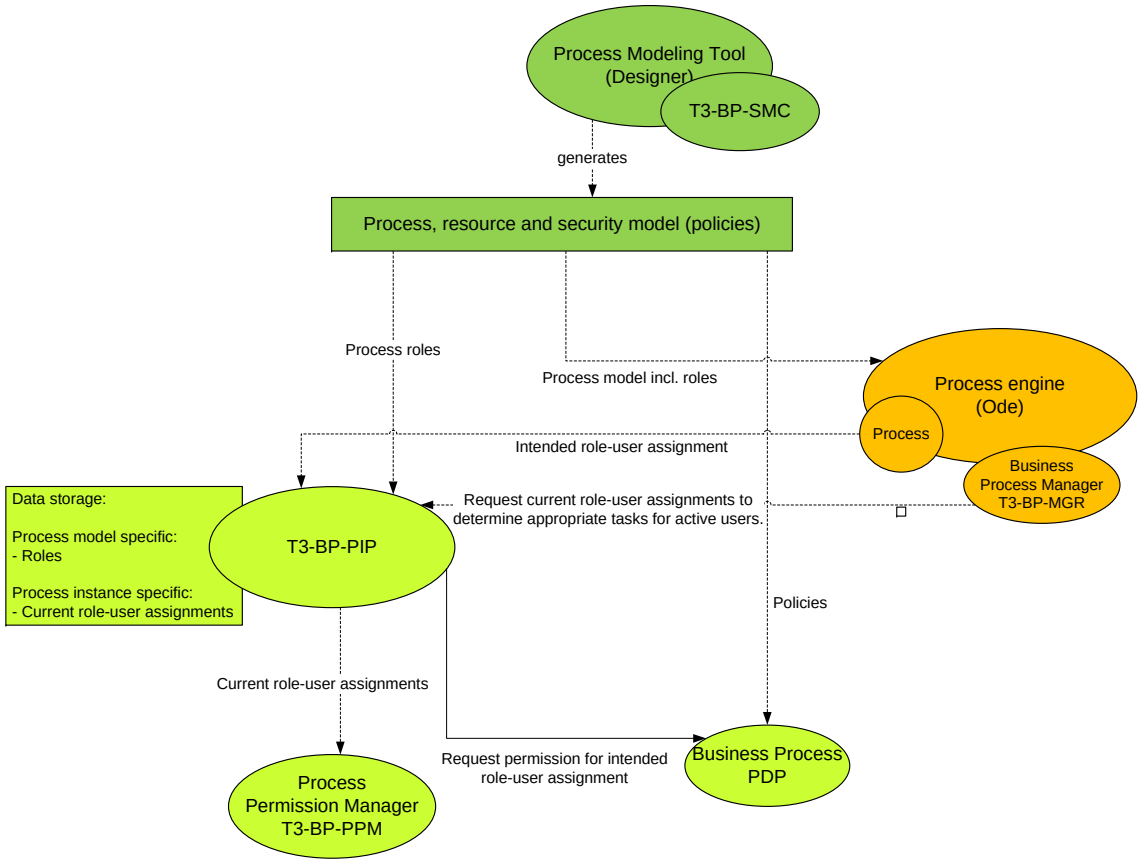


**Figure 5.4: Components interacting with the BP-PIP.**

```
<pep:request xmlns:pep="http://bpel.pep-rq.kit.tas3.eu/">
  <pep:instance-id>12345</pep:instance-id>
  <pep:endpoint-id>matchingService</pep:endpoint-id>
  <pep:action>match</pep:action>
  <pep:payload>
    <m:matchingRequest xmlns:m="http://services.tas3.eu/">
  <m:portfolio>...</m:portfolio>
  <m:programme>...</m:programme>
</m:matchingRequest>
  </pep:payload>
</pep:request>
```

**Figure 5.5: Example SOAP payload of a request to the T3-PEP-BPQ, illustrating the XML structure**

## 5.5 Delegation and Separation of Duty

No further changes to Intalio|Tempo are necessary in order to support delegation. As Tempo requests role assignments from the BP-PIP when it creates the list of open tasks for a user, delegations of process roles become effective immediately. We provide an interface where users can see their current process roles and cause delegation of roles, which will become a part of the dashboard. We provide the necessary backend functionality as part of the BP-PIP. Process designers can specify delegation policies for process roles in the BP-CLIENT component, together with the role assignment policies.

Just like binding of duties for roles, separation of duty currently can only be implemented in an imperative way: The process designer has to explicitly specify the assignment of individuals to tasks in the process model itself, including computation to determine an assignment that fulfils the SoD constraints. But this may be error-prone for large process diagrams or complex SoD constraints, as the constraints themselves are not explicitly visible. Thus, a better option is to explicitly specify the SoD constraints and to enforce these explicit constraints during execution.

Conflicts between global roles are defined together with these roles (Figure 5.6 and Figure 5.7) and is valid for the entire process. We consider also local roles in relationship to security annotations of the business process model. There can happen conflicts involving local roles. A SoD conflict relation between two roles means that the same person may not be assigned to both roles. The SoD policy is stored on the BP-PIP when being deployed and checked when assigning or delegating process roles.

Again, no further changes to the BPEL execution engine or to the workflow engine are necessary. We enhance the tool for the definition of process roles with the possibility to specify SoD constraints between roles. The major part of the implementation concerns the context store: On each request for role assignment or delegation, it needs to check whether the specified assignee or delegate is already assigned to a conflicting role. This is handled via the BP-PIP for storing the context and the BP-PEP for coordination.

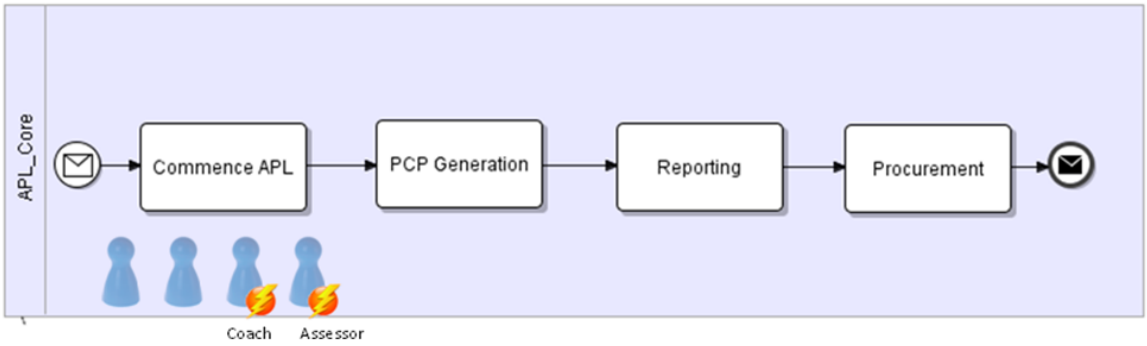


**Figure 5.6: Roles in the top-level process with an indication of SoD conflict relationships**

## 5.6 Reaction to Security Violations

Processes must be able to detect error conditions due to security violations and recover from them (requirement D3.1-R.9). First, this means that if the PEP inside the BPEL execution engine detects security violations, it must communicate them to the process. It does so by throwing a fault. A fault in WS-BPEL has a type identified by a qualified name (QName). There will be one error type for security violations occurring at normal web-service call on the application level, and one for each kind of security-specific calls, like the assignment of users to roles. The only information that can be implied from the fault is that the request was denied. Second, the process must be able to react to such faults. This is done by including fault handlers in the BPMN diagram in Intalio Designer. They are translated into WS-BPEL fault handlers. When a fault occurs, this handler executes. It can contain arbitrary activities necessary to recover from the fault, e.g. retrying the request with other parameters or initiating a break-the-glass procedure.

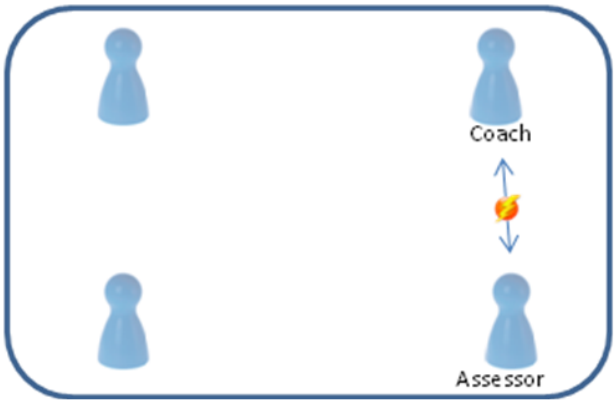


**Figure 5.7: Definition of SoD conflicts**

As fault handling is a standard WS-BPEL mechanism, no further implementation is required to handle faults. Our implementation of the BP-PIP will throw a fault when a role assignment is not allowed. Components in the TAS$^3$ security architecture are likely to create faults in error situations as well, e.g. when access to a repository is denied.

## 5.7 Administration of identity tokens

It must be possible to specify on behalf of whom a process is acting (requirement D3.1-R.10). For the calls to external systems that a process will be making during its lifetime, it can act for different people in each case. Business processes (like any web-services) will use tokens when acting on behalf of the users participating in them (cf. D2.1 [23], section 4.2).

We want to use annotations in the process model to automatically determine which token to use for outgoing calls if there are several users. This serves as a first approach to distinguish different parts in a process that require different levels of security. The annotation will either be on entire BPMN pools or on sub-processes in a BPMN diagram and consist of a process role. These annotations are transformed into a business process policy available at runtime. When a call to an external service requires an identity token, the identity mapper service will automatically be invoked to exchange the existing token into one usable by the external service, and this token will then be used in the call.

## 5.8 Structural Changes of the Process Flow

For process adaptation the structural changes of the process flow starts at the business process modelling level and then transforms the changed model to the execution level of business processes. With that, the adaptation relates to the schema level. In the next step the migration of running processes takes place, those processes, that match the old schema and the planned change is possible. We perform the following steps:

- Adapt the process by using change operations on the BPMN level and workflow pattern libraries.
- Transform the change operations and changed workflow to the execution level by

  Mapping to BPEL (on a schema level, e.g. using the Intalio mapper),

  Migrating process instances: investigate the state of the instances, select instances affected, and adapt the running instances of the process.

Structural process adaptation uses a repository of process patterns. These patterns support the choice of suitable patterns to change the process schema. There will be selected alternatives for structural changes, and automatic adaptations are supported if possible. But in general the adaptation component only proposes alternatives for structural changes. Making the decision will be a manual task by a process administrator. Figure 5.8 shows the adaptation architecture with modelling and execution level.

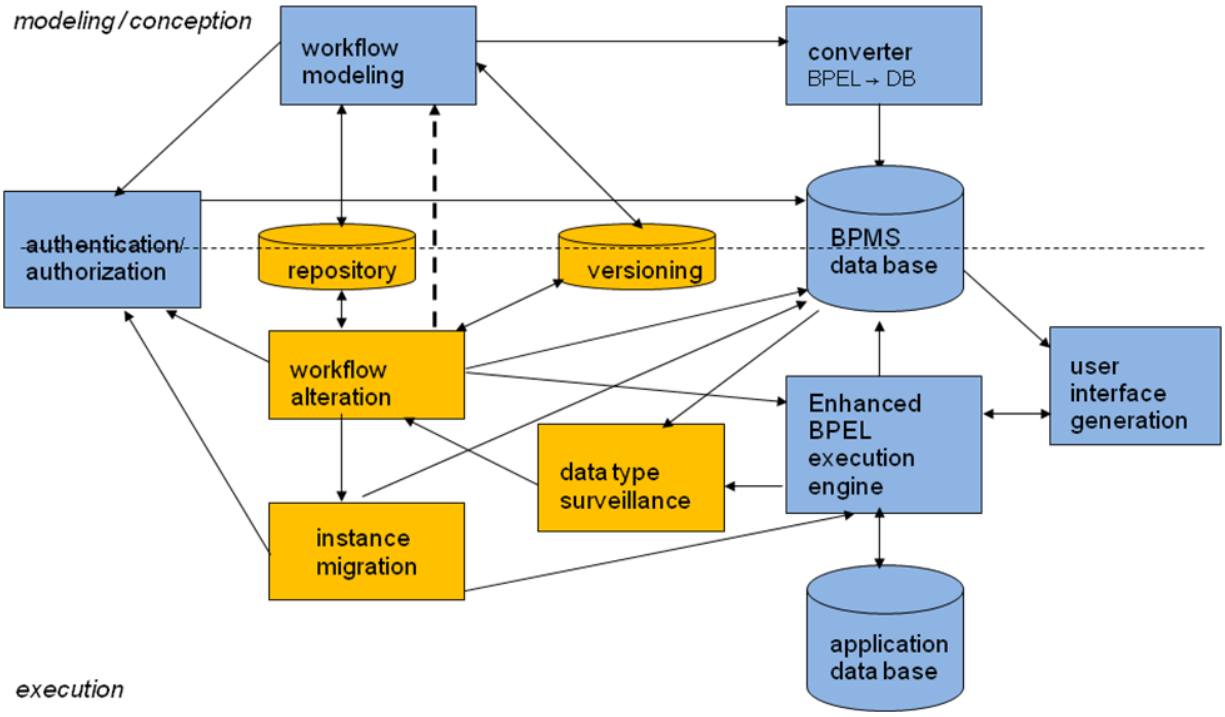


**Figure 5.8: Adaptability architecture of a WFMS**

Reasons for process adaptation are many-fold. A driving force for process changes is the change of data which is involved in the business process. Ad-hoc processes, i.e., processes that are not fully designed, e.g. because there are many users with very specific requirements regarding the process, require to enhance the process during execution. Often these requirements result from the specific data that is needed, e.g. on a special data type like a single value vs. a set of entities or aggregates of those entities, see [93]. Another reason is exceptions caused by violation of security rules when accessing data. A possible reaction could be to, say, choose other data that is accessible to the current user but having a lower security level, e.g. anonymised data. Another possible reaction would be to provide an adequate certificate to the user. In both cases, a process adaptation typically is required. Other exceptions like "specific data is not available" could be a reason to change an active process as well. In today's business processes, they usually simply fail when such unexpected events occur.

Because processes and the functional organisation of their task composition often depend on the underlying data, we have investigated how data changes influence process structure changes. We propose a method to use data dependencies to detect if a process needs to be changed. If applicable, adequate adaptations of the process schema are proposed [94]. The enhanced system architecture is shown in Figure 5.9. The data type surveillance component is responsible to detect changes of the data types that are relevant for the process structure. The component works on the database of the business process management system triggering the workflow alteration component on specific change events. The component then checks if an adaptation of the process schema becomes necessary. In this case, the component proposes workflow adaptations using the repository of workflow patterns.

Further results of ours [77] are a formal model of adaptation operations at the BPMN level and their relationship to the BPEL process model. On the instance layer it has to be checked if the modified schema is adequate for any of the running instances. A crucial issue for this is the correctness of the data flow in the process instance changed. The process instances for which the new schema is applicable must be migrated. Finally, the process instances continue executing the new process flow.

In order to be able to transform adapted processes and to manage process versions, we identified a set of adaptation functions on the BPMN and the BPEL level. As a basic model we support structural process changes with operators such as, say, AddTasks, DeleteTasks, AddSequenceFlows, SetSequenceFlowConditionExpressions. These operations are composed of reusable basic methods called *modification* (on the

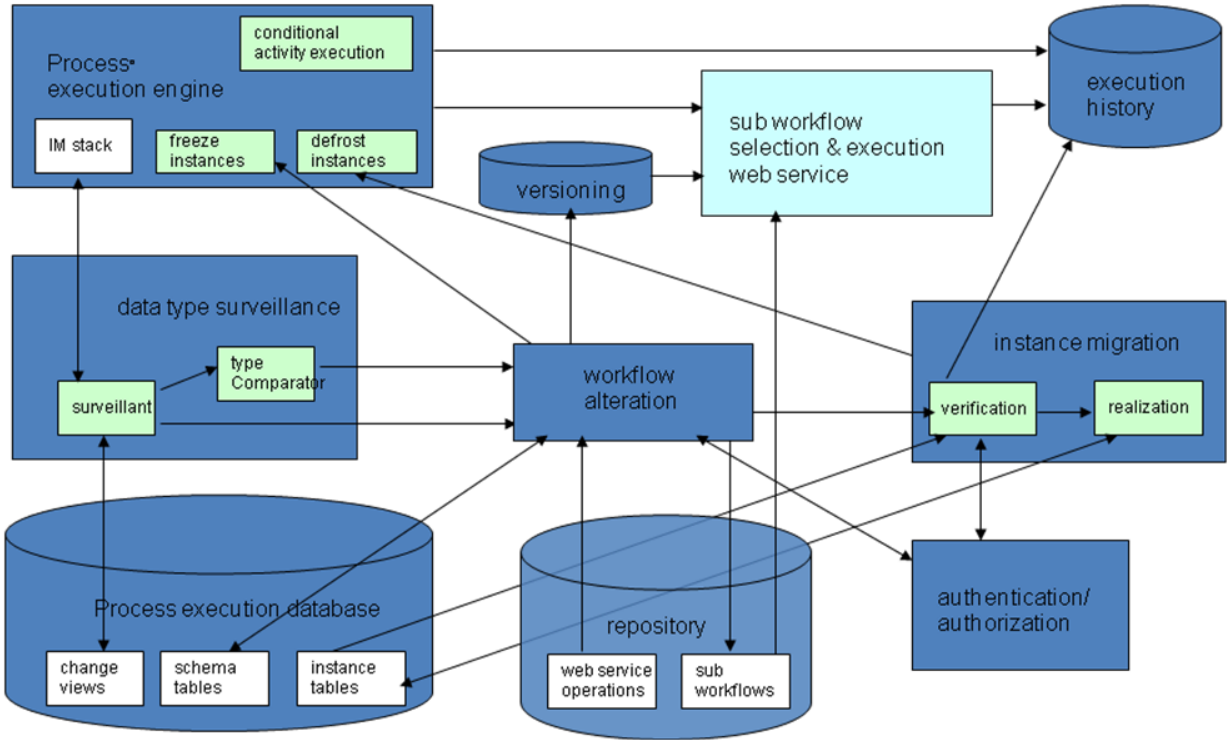


**Figure 5.9: System architecture of a business process execution engine enabling data-induced adaptation of processes**

BPMN layer) and *adaptation* (on the instance layer) *primitives*.

Figure 5.10 shows the system architecture of our structural adaptation concept and the course of actions taking place within one adaptation operation. We need an adaptation enabled process modelling tool. This can be obtained by integrating an *adaptation plugin* into an existing modelling tool for instance. The plugin or the modelling tool itself provides the user interface the process administrator is operating on. When the modification on the process modell have been performed, the model is translated to BPEL. Both the BPMN and the BPEL representation are stored in a repository, which is be accessible for the modelling tool and the process execution engine.

The adaptation plugin calls the *adaptation component* and notifies it on the adapation operation to be performed. Depending on the specific operation the adaptation component manages the adapted execution of the corresponding data flow analysis, the execution state verification and the instance migration itself. During the performance of these validations the process execution engine has to be in a consistent state. Therefor the engine is being stopped beforehand.

Before the irrevocable instance migration takes place, the adaptation component informs the process administrator about the results of the integrity checks and let him decide, which instances he really wants to migrate. After migration the process engine can continue the execution of the process instances.

These results of adaptation support will serve as a basis for more flexible and adaptable business processes in TAS$^3$. Next steps will be to investigate in more detail the influence of security specifications of parts of the process, i.e. taks, suprocesses, web services, data and actors, to the validity of process change. Especially, the security-guided choice of web services is of interest. E.g., if a web services guarantees a higher level of security but delivers the required data or functionality, the user may want to use that one. A challenge results from this kind of flexibility and tolerance, for which we are planning to use a semantic specification of the security of the process and its parts.

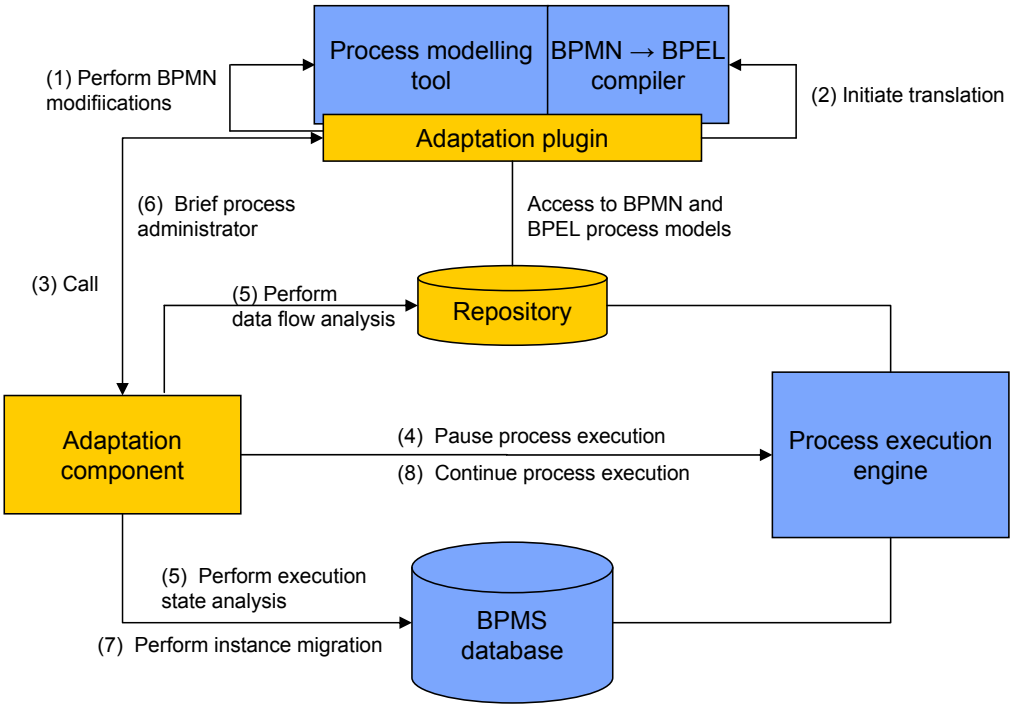


**Figure 5.10: System architecture of the structural adaptation concept**

# 6 Conclusions

This document represents the design of a business process management platform in the TAS$^3$ project up to now, i.e. the third iteration of this report. We are refining this report according the project plan in each project phases with more detailed and new results.

Regarding the requirements described in Deliverables D1.2 [5] and D1.4 [6] we have designed the various components of a platform for secure process management, i.e., a business process modelling platform and execution engine for processes in service-oriented applications, security issues of business processes, and secure adaptability of business processes. The components have to interoperate with each other and have to be integrated in the TAS$^3$ security architecture. To accomplish this, we developed the architecture by refining and amending the overall TAS$^3$ architecture.

One of our goals is to provide a platform that enables the partners to model secure business processes in the application scenarios and run the business processes modelled. To this end, there have been structured training sessions for business process modelling using the Intalio tools, and we modelled a real-world business process from the employability scenario, the Kenteq APL process, in cooperation with modelling experts and domain experts. On this basis, we have been able to model more complex processes in the employability area requiring flexibility and in particular security specifications. In the 2nd project year there was the first Nottinham student placement demonstrator, which has been enhanced in the 3rd project year to a more comprehensive second version. Additionally, in the third project year we cooperated with several project partners to realize a Kenteq Mass Layoff Demonstrator. All of these demonstrators are described in detail in D3.3 [3]. We have already used the results of the 2nd project year demonstrator to validate our conceptual design and to adapt it according to the validation results. The same procedure has happened in the 3rd project year with validating these demonstrators. To this end, we have been able to check various process and design alternatives and have investigated the further integration of business processes with the TAS$^3$ security levels.

The main focus of this work package is on security of processes: Which concepts will be required, what is provided by standards, which enhancements do we need? The results are placed on three layers: to enhance the process modelling tool with features to specify security in a more comprehensive way at the business layer. On the next layer: to provide a transformation of these security-related enhancements of the business process to policies and parameters to configure security components in the TAS$^3$ security infrastructure. Finally: to design and establish concepts for security enforcement in business processes resulting in a secure business process management system architecture.

All these layers provide an enhanced and more detailed conceptual design and a secure business process management system from modelling, process adaptation and configuration to secure execution of business processes. We achieved results in detailing the business-process specific security mechanisms, in particular we have introduced a business process security policy and its relationship to the other components. With respect to adaptability we have a concrete comprehensive overview about the techniques of our adaptation approach and added mechanism to adapt processes with use of process fragments and provided conceptual design of adapting processes on the instance level. Our respective research strongly focuses on the relationship between adaptation and process security concepts. Finally, our approach for modelling security for business processes has started from a basic approach and used iteratively the experiences with the application demonstrators for evolving. The security annotations of process models now build a comprehensive set of security rules, and our approach for transforming the specifications to the enforcement level. To provide better usability, we have introduced an ontology-based annotator supporting the modellers in specifying security annotations.

Based on experiences with pilot applications and iterative design of BP-specific security components, we have derived a modular architecture of a secure BPMS. It is based on established reference models in the fields of BPMS (WfMC workflow reference model) and access control (XACML reference architecture). It requires comparatively few changes to legacy components when integrating them into the architecture. We see this architecture as an important driver for the integration of existing solutions.

A main issue of WP3 in the last project period has been to implement the advanced approaches of the

conceptual design in a software version which validates our conceptual design against the TAS$^3$ security requirements and against the applications. We have integrated the process management platform in the TAS$^3$ architecture in close cooperation with WP8 and with pilots of WP9 in three demonstrators. In the last project period we have investigated how Break-the-Glass can be supported in business processes. To this end, we developed a concept, and enhanced the security annotation language (showing its flexibility and openness). We then demonstrated how BTG can take advantage of BP technology in providing a more comfortable way for users to modelling of BTG scenarios. This results from configuring the secure BPMS to handle BTG and obligations of a an application process with pre-defined process fragments. Taking this approach, we have realized an example scenario of BTG in secure business processes in the e-health domain.

# Bibliography


[1] J. Müller and K. Böhm, "The Architecture of a Secure Business-Process-Management System in Service-Oriented Environments," in *Proceedings of the 9th IEEE European Conference on Web Services*, 2011.

[2] J. Mülle (ed.), "Open source software and documentation implementing the design – software for secure business processes." TAS3 Deliverable 3.2, Final Version, October 2011.

[3] ——, "Integration with TAS$^3$ trust applications of employment and eHealth processes." TAS3 Deliverable 3.3, 3rd Iteration, October 2011.

[4] S. Kellomäki (Ed.), "TAS3 Architecture," TAS3 Deliverable 2.1, final, July 2011.

[5] S. Gürses and N. Zannone (Eds.), "Requirements Assessment Report, TAS3 Deliverable 1.2, Rev. 1."

[6] G. Montagnon (Ed.), "Design requirements, TAS3 Deliverable 1.4," 2010.

[7] OASIS, "Web Services Business Process Execution Language Version 2.0," OASIS Standard., April 2007. [Online]. Available: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

[8] Object Management Group, "Business Process Modeling Notation, V1.2," OMG Available Specification, January 2009. [Online]. Available: http://www.omg.org/spec/BPMN/1.2/PDF/

[9] Workflow Management Coalition, "Architecture of a Workflow Management System," October 2008. [Online]. Available: http://www.wfmc.org/

[10] Intalio, "Intalio—BPMS - Business Process Management Suite." [Online]. Available: http://community.intalio.com/

[11] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "Web Services Description Language (WSDL) 1.1," W3C Note, World Wide Web Consortium, [Online], 15 March 2001. [Online]. Available: http://www.w3.org/TR/wsdl

[12] Active Endpoints, Adobe, BEA, IBM, Oracle, SAP AG, "WS-BPEL Extension for People," June 2007.

[13] Active Endpoints, Adobe, BEA, IBM, Oracle, SAP AG, "Web Services Human Task (WS-HumanTask), Version 1.0," June 2007.

[14] Intalio, "Intalio Business Process Management." [Online]. Available: http://www.intalio.com/products/bpm/

[15] ——, "Intalio Designer." [Online]. Available: http://www.intalio.com/products/bpm/community-edition/designer/

[16] ——, "Intalio Server." [Online]. Available: http://www.intalio.com/products/bpm/community-edition/server/

[17] Apache, "Apache Ode (Orchestration Director Engine)." [Online]. Available: http://ode.apache.org/

[18] Intalio, "Intalio Tempo." [Online]. Available: http://tempo.intalio.org

[19] OASIS, "Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)," OASIS Standard Specification., February 1 2006. [Online]. Available: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

[20] Wikipedia, "Xforms." [Online]. Available: http://en.wikipedia.org/wiki/XForms

[21] TIBCO, "TIBCO General Interface." [Online]. Available: http://developer.tibco.com/gi/

[22] G. Montagnon (Ed.), "Design requirements, TAS3 Deliverable 1.4," 2009.

[23] S. Kellomäki (Ed.), "TAS3 Architecture," TAS3 Deliverable 2.1, 1st Iteration, June 2009.

[24] R. Geneva and T. Debevoise, *The Microguide to Process Modeling in BPMN*. BookSurge Publishing, July 2007.

[25] B. Claerhout (Ed.), "Pilots Specifications and Use Case Scenarios, TAS3 Deliverable 9.1, Rev. 1."

[26] J. Müller, J. Mülle, S. von Stackelberg, and K. Böhm, "Secure Business Processes in Service-Oriented Architectures - a Requirements Analysis," in *Proceedings of the 8th IEEE European Conference on Web Services (ECOWS 2010)*, 2010, pp. 35–42.

[27] D. Chadwick (Ed.), "Design of Identity Management, Authentication and Authorization Infrastructure, e, TAS3 Deliverable 7.1, Rev. 1."

[28] T. Consortium, "State of The Art, TAS3 Deliverable 1.1, Rev. 1."

[29] I. Weber, J. Haller, and J. Mülle, "Automated derivation of executable business processes from choreographies in virtual organisations," *Int. Journal of Business Process Integration and Management, Inderscience*, vol. vol. 3, pp. 85–95, 2008.

[30] J. Dehnert and W. van der Aalst, "Bridging the gap between business models and workflow specifications," *Int. J. Coop. Inf. Syst.*, vol. 13, pp. 289–332, 2004.

[31] D. Martin, M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuiness, B. Parsa, T. Payne, M. Sabou, M. Solanki, N. Srinivasan, and K. Sycara, "Bringing Semantics to Web Services: The OWL-S Approach," *in: the First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*, 2004.

[32] D. Roman, U. Keller, H. Lausen, J. de Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel, "Web Service Modeling Ontology," *Applied Ontology*, vol. 1, pp. 77–106, 2005.

[33] R. Lara, A. Polleres, H. Lausen, D. Roman, J. de Bruijn, and D. Fensel, "A Conceptual Comparison between WSMO and OWL-S. WSMO Deliverable D4.1," 2005. [Online]. Available: http://www.wsmo.org/2004/d4/d4.1/v0.1/20050106/

[34] Bertino, Elisa et al., *Security for Web Services and Service-Oriented Architectures*. Springer, 2010.

[35] K. P. Peralta and A. Zorzo, "Specifying Security Aspects in UML Models," in *Proc. CEUR Modelling Security Workshop, Models 08, Toulouse*, September 2008.

[36] M. Hafner and R. Breu, *Security Engineering for Service-Oriented Architectures*. Springer Verlag, Berlin, 2009.

[37] D. Basin, J. Doser, and T. Lodderstedt, "Model Driven Security for Process-Oriented Systems," in *Proc SACMAT 03, Como, Italy*, June 2003.

[38] C. Wolter, C. Meinel, and M. Menzel, "Modelling security goals in business processes," in *Lecture Notes of Computer Science, 127*, 2008, pp. 197 – 212.

[39] M. Backes, B. Pfitzmann, and M. Waidner, "Security in business process engineering," in *Proc. of the Int. Conf. on Business Process Management*, ser. BPM'03. Heidelberg: Springer, 2003, pp. 168–183.

[40] A. Rodríguez, E. Fernández-Medina, and M. Piattini, "A BPMN extension for the modeling of security requirements in business processes," *Trans. Inf. Syst. – IEICE*, vol. E90-D, pp. 745–752, March 2007.

[41] C. Wolter and A. Schaad, "Modeling of task-based authorization constraints in BPMN," vol. Volume 4714/2007, pp. 64–79, 2007.

[42] C. Wolter, M. Menzel, and C. Meinel, "Modelling security goals in business processes," in *Modellierung'08*, 2008, pp. 197–212.

[43] M. Menzel, I. Thomas, and C. Meinel, "Security requirements specification in service-oriented business process management," in *ARES*. IEEE Comp. Society, 2009, pp. 41–48.

[44] Object Management Group, "Business Process Model and Notation, V2.0," OMG Available Specification, January 2011. [Online]. Available: http://www.omg.org/spec/BPMN/2.0/PDF

[45] J. Alhadeff and B. V. Alsenoy, "Requirements: Privacy, governance and contractual options, TAS3 Deliverable 6.1," 2009.

[46] OASIS, "eXtensible Access Control Markup Language tc v2.0 (XACML)," February 2005.

[47] D. Hollingsworth, "The Workflow Reference Model," Workflow Management Coalition, WfMC Specification TC00-1003, 1995.

[48] S. Sendall and W. Kozaczynski, "Model transformation: The heart and soul of model-driven software development," *IEEE Softw.*, vol. 20, pp. 42–45, September 2003.

[49] C. Wolter, A. Schaad, and C. Meinel, "Deriving XACML policies from business process models," in *WISE Workshops*, ser. LNCS, vol. 4832. Springer, 2007, pp. 142–153.

[50] C. Wolter and et al., "Model-driven business process security requirement specification," *Journal of Systems Architecture*, vol. 55, pp. 211–223, 2009.

[51] A. Rodríguez, "Semi-formal transformation of secure business processes into analysis class and use case models: An MDA approach," *Inf. Softw. Technol.*, vol. 52, pp. 945–971, September 2010.

[52] D. Basin, J. Doser, and T. Lodderstedt, "Model driven security: From UML models to access control infrastructures," *ACM Trans. Softw. Eng. Methodol.*, vol. 15, pp. 39–91, January 2006.

[53] M. Richters and M. Gogolla, "Validating UML models and OCL constraints," in *Proc. UML2000 - The Unified Modeling Language. Advancing the Standard. 3rd Int.Conf.*, vol. LNCS 1939. Springer, 2000.

[54] J. Mülle (ed.), "Open source software and documentation implementing the design – software for secure business processes." TAS3 Deliverable 3.2, 1st Iteration, December 2009.

[55] Q. Reul (Ed.), "Lower Common Ontology, TAS3 Deliverable 2.3, Rev. 1."

[56] P. Spyns, Y. Tang, and R. Meersman, "An Ontology Engineering Methodology for DOGMA," *Journal of Applied Ontology*, vol. 3, pp. 13–39, 2008.

[57] Q. Reul (Ed.), "Common Upper Ontology, TAS3 Deliverable 2.2, Rev. 1."

[58] S. Jablonski and C. Bussler, *Workflow Management: Modeling Concepts, Architecture and Implementation*. International Thomson Computer Press, 1996.

[59] A. Charfi and M. Mezini, "Aspect-Oriented Workflow Languages," in *On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE*, 2006.

[60] J. Mülle, S. von Stackelberg, and K. Böhm, "Modelling and transforming security constraints in privacy-aware business processes," in *Proc. SOCA'2011, Irvine, California, to be published*, 2011.

[61] W. van der Aalst and K. van Hee, *Workflow Management: Models Methods and Systems*. MIT Press, 2002.

[62] J. Müller, M. Kavak, and K. Böhm, "A Graphical Audit Facility for Data Processing and its Evaluation with Users," in *submitted*, 2011, http://www.ipd.kit.edu/ muelle/mueller-wosec-2011.pdf.

[63] J. Müller and K. Böhm, "Using Federated Identity Management in a Business-Process-Management System – Requirements, Architecture, and Implementation," Karlsruhe Reports in Informatics 2011-28, September 2011, http://www.ubka.uni-karlsruhe.de/eva/index.html (to appear).

[64] S. Enge, "Access Control in Adaptive Workflow Management Systems." (in German), diploma thesis, University of Karlsruhe, Germany, August 2007.

[65] W. van der Aalst, "The application of petri nets to workflow management," *Journal of Circuits, Systems and Computers*, vol. 7, no. 1, pp. 1–45, 1998.

[66] M. Reichert, "Dynamische ablaufnderungen in workflow-management-systemen." dissertation, University of Ulm, 2000.

[67] W3C, "OWL 2. Web Ontology Language." 2009. [Online]. Available: http://www.w3.org/TR/owl2-overview/

[68] ——, "OWL-S. Semantic Markup for Web Services." 2004. [Online]. Available: http://www.w3.org/Submission/OWL-S/

[69] D. Roman, H. Lausen, and U. Keller, "The Web Service Modeling Ontology WSML." 2006. [Online]. Available: http://www.wsmo.org/TR/d2/v1.3/

[70] The WSML working group members, "The Web Service Modeling Language WSML." 2008. [Online]. Available: http://www.wsmo.org/wsml/

[71] B. Andrachnik, "Transformation of BPMN Security Annotations using Workflow Patterns." (in German), bachelor thesis, Karlsruhe Institute of Technology, Germany, September 2010.

[72] M. Racke, "Flexible and ad-hoc changeable workflow realization with BPEL4WS." (in German), diploma thesis, University of Karlsruhe, Germany, May 2006.

[73] J. Mülle (ed.), "Integration with TAS$^3$ trust applications of employment and eHealth processes." TAS3 Deliverable 3.3, 2nd Iteration, December 2010.

[74] ——, "Integration with TAS$^3$ trust applications of employment and eHealth processes." TAS3 Deliverable 3.3, 1st Iteration, December 2009.

[75] Object Management Group, "Business Process Modeling Notation, V1.1," OMG Available Specification, January 2008. [Online]. Available: http://www.omg.org/spec/BPMN/1.1/PDF/

[76] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana, "Business Process Execution Language for Web Services, Version 1.1," May 2003. [Online]. Available: http://www-128.ibm.com/developerworks/library/specification/ws-bpel/

[77] T. Haberecht, "Structural adaptation of BPMN/BPEL-Workflows and integration with the Intalio BPMS." (in German), diploma thesis, University of Karlsruhe, Germany, March 2009. [Online]. Available: http://www.thorsten-haberecht.de/Adaption.pdf

[78] D.F. Ferraiolo and D.R. Kuhn, "Role Based Access Control," in *Proc. 15th National Computer Security Conference, Oct 13-16*, 1992, pp. 554–563.

[79] R. S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman, "Role-Based Access Control Models," *IEEE Computer*, vol. 29(2), pp. 38–47, 1996.

[80] R. Sandhu, D. Ferraiolo, and R. Kuhn, "The NIST Model for Role-Based Access Control: Towards a Unified Standard," in *ACM 5th Workshop on Role Based Access Control*, 2000.

[81] International Committee for Information Technology Standards, "Information technology - role based access control, ansi/incits 359-2004 (standard)," International Committee for Information Technology Standards.

[82] T. Priebe, W. Dobmeier, C. Schläger, and N. Kamprath, "Supporting Attribute-based Access Control in Authentication and Authorization Infrastructures with Ontologies," *Journal of software: JSW*, vol. 2, no. 1, pp. 27–38, Februar 2007. [Online]. Available: http://epub.uni-regensburg.de/6423/

[83] R. K. Thomas, "Team-based Access Control (TMAC): A Primitive for Applying Role-based Access Controls in Collaborative Environments," in *Proc. RBAC 1997, Fairfax, Virginia, USA*, 1997.

[84] C. K. Georgiadis, I. Mavridis, G. Pangalos, and R. K. Thomas, "Flexible Team-based Access Control using Contexts," in *SACMAT '01: Proceedings of the Sixth ACM Symposium on Access Control Models and Technologies*. New York, NY, USA: ACM, 2001, pp. 21–27.

[85] V. Atluri and W. kuang Huang, "An Authorization Model for Workflows," in *Proc. of the 4th European Symposium on Research in Computer Security: Computer Security*, 1996.

[86] R. K. Thomas and R. S. Sandhu, "Task-Based Authorization Controls (TBAC): A Family of Models for Active and Enterprise-Oriented Autorization Management," in *Proceedings of the IFIP TC11 WG11.3 11th International Conference on Database Securty XI: Status and Prospects*. London, UK, UK: Chapman & Hall, Ltd., 1998, pp. 166–181.

[87] Y. Asselborn, "Design and Implementation of Permission Handling for a Workflow-Management System." (in German), master thesis, Karlsruhe Institute of Technology, Germany, 2011.

[88] Erik Rissanen (ed.), "eXtensible Access Control Markup Language (XACML) Version 3.0," Committee Specification 01, August 2010. [Online]. Available: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cs-01-en.pdf

[89] ——, "XACML v3.0 Administration and Delegation Profile Version 1.0," Committee Specification 01, August 2010. [Online]. Available: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-administration-v1-spec-cs-01-en.pdf

[90] E. Hammer-Lahav, "The OAuth 1.0 Protocol," RFC 5849 (Informational), Internet Engineering Task Force, Apr. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5849.txt

[91] D. W. Chadwick, W. Xu, S. Otenko, R. Laborde, and B. Nasser, "Multi-Session Separation of Duties (MSoD) for RBAC," in *Proc. First International Workshop on Security Technologies for Next Generation Collaborative Business Applications (SECOBAP'07), Istanbul, Turkey*, 2007.

[92] J. Crampton and H. Khambhammettu, "Delegation in role-based access control," *Int. J. Inf. Secur.*, vol. 7, pp. 123–136, March 2008.

[93] J. Mülle, K. Böhm, N. Röper, and T. Sünder, "Building Conference Proceedings Requires Adaptive Workflow and Content Management," in *Proc. 32nd Intl. Conf. on Very Large Data Bases, Seoul, Korea*, 2006.

[94] D. Weingardt, "Extending a WFMS with Data-Induced Adaptability." (in German), diploma thesis, University of Karlsruhe, Germany, March 2008.

# Glossary

- B4P or BPeL4People: enhancement of the BPEL standard to support human activities
- BPEL: Business Process Execution Language
- BPM: Business Process Modelling
- BPMN: Business Process Modelling Notation
- BP-PAP: Business Process Policy Administration Point
- CVS: Credential Validation Service
- DIS: Delegation Issuing Service
- DPM: Delegated Permissions Manager
- IA-PIP: Instance-Attribute Policy Information Point
- IR-PIP: Instance-Role Policy Information Point
- OWL: Web Ontology Language
- OWL-S: OWL based web service ontology
- PCP: Personal Competency Profile
- PEP: Policy Enforcement Point
- PIP: Policy Information Point
- PMF: Process Modelling Framework
- QC: Quality Controller
- SoD: Separation of Duties
- SP: Service Provider
- TN: Trust Network
- WSMO: Web Service Modelling Ontology

# Annex A: Security Modelling Language for Business Processes

Our security language is embedded in BPMN 2.0 as structured text annotations. The aim is to represent security constraints in business processes declaratively. Having identified all security requirements of business processes of two complex real-world scenarios in the employability domain in SOA environments, our language covers all these issues, i.e., the requirements from the previous section. This chapter completely specifies our language. We group our language primitives into different categories, namely authorization, authentication, auditing, data and message flow security, and security- and trust-specific user involvements. The first categories are well-known in the security literature, but not fully specified in existing approaches for business processes ([39], [40], [41], [42], [43]). Additionally, we extend authorization constraints by controlling the rights for process adaptations. Annotations for security- and trust-specific user involvements are a new notion, aiming to support user centricity.

Annotations have the following generic structure:

$$\ll \textit{AnnotationTerm}\text{: } \textit{list(parameterName}\text{=“}\textit{value}\text{”)} \gg$$

For example, ≪BoD≫ is a simple annotation term, specifying a binding of duty (BoD) constraint. The security annotations require no, one, or many parameter-name/value-pairs, dependent on the term. Some are optional in certain cases. If nothing else is said in the following, the parameter values are “string” data types.

Each annotation term is defined for a particular set of BPMN 2.0 elements, e.g., for activities, lanes, data objects, data stores, or message flows. According to the BPMN standard, an activity can be a simple task or a sub-process.

We now describe the annotation terms for authorization, authentication, auditing, data and message flow security, and security-specific user involvements in detail.

## 6.1 Authorization Constraints

The authorization constraints (Table 6.1) specify who possesses which rights under which restrictions. The dots in Table 6.1 stand for missing parameters. To manage authorizations, we apply roles, i.e., we specify access rights for role holders.

| Authz Constraint | Syntax |
|---|---|
| Role Assignment | ≪Assignment: type=“Role” name=“$rolename”≫ |
| Mechanism Assignm. | ≪Assignment: type=“Mechanisms” ... ≫ |
| User Assignment | ≪Assignment: type=“User” name=“$username”≫ |
| Separation of Duty | ≪SoD: role=“$rn” number=“$nr” ... ≫ |
| Binding of Duty | ≪BoD: spec=“weak”... ≫ |
| Adaptation | ≪Adaptation: rights = “$rightsname”... ≫ |
| Task-Delegation | ≪T-Delegation: object=“$activityname”... ≫ |
| Data-Delegation | ≪D-Delegation: rights = “$rightsname”... ≫ |

**Table 6.1: Overview of Authorization Constraints**

### Role assignment

#### Description

The term

$$\ll \textit{Assignment: } \textit{type} = \text{“}\textit{Role}\text{”} \ \textit{name} = \text{“}\$\textit{rolename}\text{”} \gg$$

assigns a role to an activity, to a group of activities, to pools, or to lanes. It means that users with the given role are allowed to execute activities.

| Annotation | Activities | Gr. of Activ. | Pools&Lanes | Data | Messageflows | Events |
|---|---|---|---|---|---|---|
| Role Assignment | X | X | X | - | - | - |

**Table 6.2: Allowed BPMN elements for *Role Assignment***

| Attribute name | Value facet | Obligatory | Description | BPMN Element |
|---|---|---|---|---|
| role | *String* | yes | specifies a role name | Activity and Gr. of Activities |
| role | *String* | no | specifies a role name | Pools and Lanes |

**Table 6.3: Parameter for *Role Assignment***

If no role name is specified, the annotation implies that the pool or lane name determine the role name. Clearly, this option is only possible for the annotation of pools or lanes and means that users holding role $poolname$ or $lanename$ are allowed to perform tasks of the annotated pool or lane. In particular cases, if the role name differs from the name of pools and lanes, pools and lanes are annotated with a role name. In these cases, role is an obligatory parameter. A role assignment with the parameter/value-pair $role =$ "$\$rolename$" is typically used for the annotation of activities or group of activities, indicating that role holders of $rolename$ have to perform annotated activities. For simplification, each BPMN element can be annotated only by one role name. In case that many roles are required, the process modeller adapts the role name accordingly (e.g., by introducing an additional role name representing all required roles).

#### Syntax

$$\ll \textit{Assignment: } type = \text{“}Role\text{”}\ role = \text{“}\$rolename\text{”} \gg$$

#### Example

$$\ll \textit{Assignment: } type =\!\!'' \ Role''\ role =\!\!'' \ Leaner'' \gg$$

Figure 6.1 shows two role assignments for lane “placement coordinator” and lane “learner” for a student-placement scenario. This means that lane “placement coordinator” and lane “learner” represent roles.

#### Transformation

| BP Security Policy | Process Adaption | Parameters for BP security components |
|---|---|---|
| x | | x |

## Assignment Mechanism

#### Description

An Assignment Mechanism specifies the assignment of work items to role holders. We specify a mechanism to be used as follows:

$$\ll \textit{Assignment: type=“Mechanism”}\ mechanism = \text{“}\$mechanismname\text{”} \gg$$

The $mechanismname$ indicates the mechanisms to be used. A mechanism can be specified for activities, group of activities, and for pools and lanes.

#### Syntax

$$\ll \textit{Assignment: type=”Mechanism”}\ mechanism =\!\!'' \ \$mechanismname\text{”} \gg$$

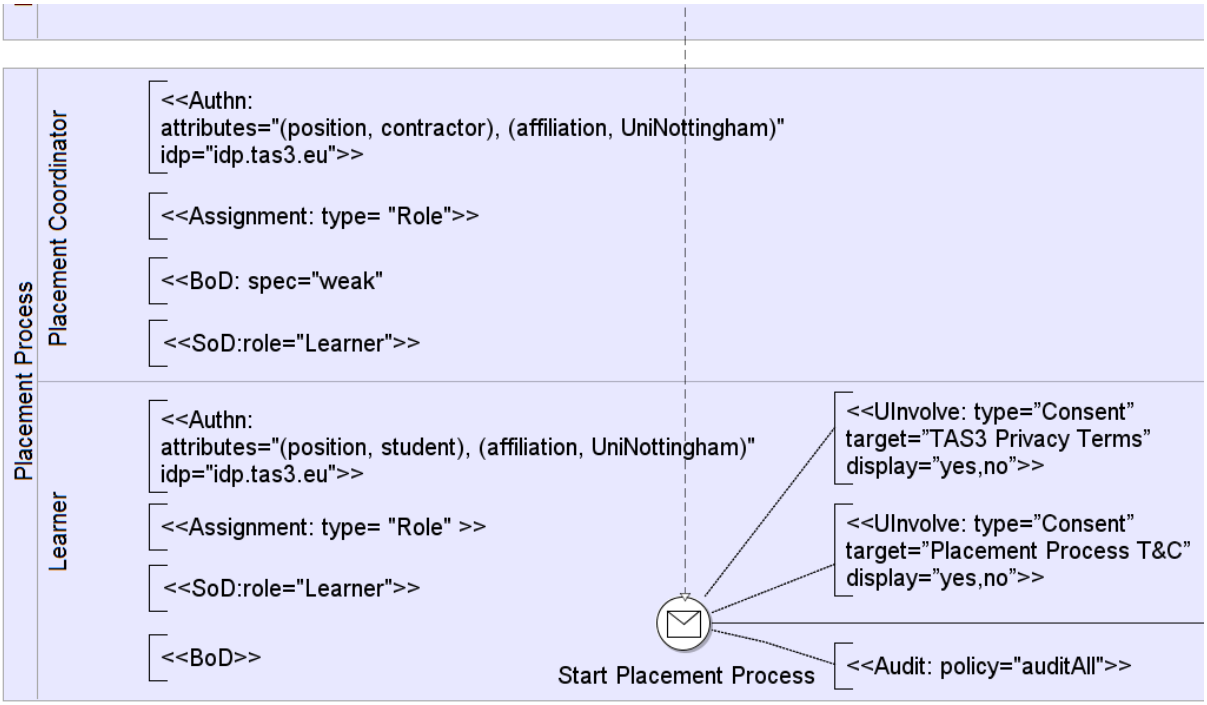


**Figure 6.1: Example Security Annotations for Process Start Event**

| Annotation | Activities | Gr. of Activ. | Pools&Lanes | Data | Messageflows | Events |
|---|---|---|---|---|---|---|
| Assignment Mechanism | X | X | X | - | - | - |

**Table 6.4: Allowed BPMN elements for *Assignment Mechanism***

#### Example

$$\ll \textit{Assignment: type="Mechanism"}\ mechanism =''\ MostExperiencedUser" \gg$$

For example, an employability company preferably assigns clerks to a customer case who have already gained experience with that applicant/customer. By supporting the specification of different assignment schemes (e.g., "assignment to role holders with the most experience", or "shortest execution time"), allocation of role holders becomes more flexible, as opposed to role holders picking up items from a work list. We facilitate the specification of assignment schemes in the process model, because they are application-specific.

#### Prerequisites

*Role assignment* annotation.
A mechanism can be specified for any BPMN elements with a corresponding role assignment. As roles are not part of BPMN, we allow the indirect assignment of mechanisms to roles in BPMN diagrams as follows: (1) pools or lanes represent roles (expressed by a role assignment) and are annotated with a mechanisms assignment accordingly. (2) activities or group of activities that have to be performed by a role (specified by an annotated role assignment) can be annotated additionally with a mechanism assignment. Such

| Attribute name | Value facet | Obligatory | Description |
|---|---|---|---|
| mechanism | *String* | yes | Specifies a mechanism name |

**Table 6.5: Parameter for *Assignment Mechanism***

twofold annotations will be interpreted accordingly.

#### Transformation

| BP Security Policy | Process Adaption | Parameters for BP security components |
|---|---|---|
| | | x |

## User Assignment

#### Description

Our language also allows for direct **user assignment** (Table 6.1). A user "$username$" (process participant) is assigned explicitely to an activity or a group of activites by the annotation

$$\ll \textit{Assignment: type="User"}\ user = \text{"\$}username\text{"} \gg$$

This means that the user "$username$" has the rights to perform the annotated activities. Typically, individual users are not assigned to particular tasks because a users' absence blocks the process execution. Consequently, this assignment has to be used only for exceptional cases.

#### Syntax

$$\ll \textit{Assignment: type="User"}\ user ='' \$username'' \gg$$

| Annotation | Activities | Gr. of Activ. | Pools&Lanes | Data | Messageflows | Events |
|---|---|---|---|---|---|---|
| User Assignment | X | X | X | - | - | - |

**Table 6.6: Allowed BPMN elements for *User Assignment***

| Attribute name | Value facet | Obligatory | Description |
|---|---|---|---|
| user | *String* | yes | specifies a user name |

**Table 6.7: Parameters for *User Assignment***

#### Example

$$\ll \textit{Assignment: type="User"}\ user ='' Bob'' \gg$$

#### Transformation

| BP Security Policy | Process Adaption | Parameters for BP security components |
|---|---|---|
| x | | x |

## Separation of Duty

#### Description

The annotations **separation of duty** (SoD) and **binding of duty** (BoD) specify dependencies of process activities. They constrain authorizations for a group of activities (or for multiple instances of an activity), to be performed by different individuals (SoD) or by the same one (BoD).

The annotation

$$\ll \textit{SoD: } role ='' \$rolename''\ number = \text{"\$}value\text{"}\ threshold ='' \$value'' \gg$$

enforces separation of duty. We offer various possibilities for the specification: Without the optional parameter/value pair $role =$ "$rolename", it must be simply ensured that the annotated objects (typically

activities) must be performed by different role holders. The specification $role$ = "$\$rolename$" excludes role holders of *"rolename"* to perform the activities. With the optional parameters $number$ = "$\$value$" and $threshold$ = "$\$value$" (both of type integer) we specify the minimum number of different users (number) that have to perform activity/activities and a threshold value of the sum of activity instances a user is allowed to perform (these cardinalities are adapted from [41]).

We allow the "separation of duty" annotation (Table 6.1) for activities, groups of activities, pools, and lanes. A SoD constraint for several lanes forbids the allocation of a user to several roles. It holds for all activities of the pool or lane.

#### Syntax

$$\ll \textit{SoD: } role = \text{"}\$rolename\text{"}\ number = \text{"}\$value\text{"}\ threshold = \text{"}\$value\text{"} \gg$$

| Annotation | Activities | Gr. of Activ. | Pools&Lanes | Data | Messageflows | Events |
|---|---|---|---|---|---|---|
| Separation of Duty | X | X | X | - | - | - |

**Table 6.8: Allowed BPMN elements for *Separation of Duty***

| Attribute name | Value facet | Obligatory | Description |
|---|---|---|---|
| role | String | no | excludes roleholders from performing |
| number | Integer | no | min. number of different users to perform activity |
| threshold | Integer | no | max. number of activity instances a user is allowed to perform |

**Table 6.9: Parameters for *Separation of Duty***

#### Example

$$\ll \textit{SoD: } role = \text{"}Leaner\text{"}\ number = \text{"}2\text{"}\ threshold = \text{"}2\text{"} \gg$$

for a group of three activities.

A setting in the employability context where a SoD constraint is meaningful is as follows: A placement coordinator cannot carry out his own placement as learner (see Figure 6.1).

#### Transformation

| BP Security Policy | Process Adaption | Parameters for BP security components |
|---|---|---|
| x | | |

## Binding of Duty

#### Open issues:

Who performs the reassignment? The WFMS or the user? In the latter case: Do we need a additional parameters to restrict the reassignment to specific roles/users?

#### Description

The concept of binding of duty constraints by the term

$$\ll \textit{BoD: } spec = \text{"}weak\text{"} \gg$$

that (typically) a group of activities will be performed by the same user. Such a restriction is desirable in order to obtain clear responsibility for a related set of activities. The annotation ≪ BoD ≫ is defined for

activities, group of activities, and pools and lanes. Usually, a set of activities or a group of activities is specified by a BoD constraint. In the particular case, if only one activity is annotated, SoD constraints for loops or activity instances of a process are described. The default case is without the parameter *spec*. To avoid blocked process instances due to absence of process participants, we allow to ease BoD constraints by enabling the explicit transfer of responsibility *i.e., re-assignment* for activities. Note that we do not use the term delegation here, since delegation has a different meaning later. If the optional parameter $spec =$ "$weak$" is specified, a re-assignment of activities is possible, otherwise not.

To avoid process blocking because users are absent, we allow to ease BoD constraints by explicit transfer of responsibility, *i.e., re-assignment* of activities. The optional parameter $spec =$ "$weak$" specifies this.

#### Syntax

$$\ll \textit{BoD: } spec = \text{“}weak\text{”} \gg$$

| Annotation | Activities | Gr. of Activ. | Pools&Lanes | Data | Messageflows | Events |
|---|---|---|---|---|---|---|
| Binding of Duty | X | X | X | - | - | - |

**Table 6.10: Allowed BPMN elements for *Binding of Duty***

| Attribute name | Value facet | Obligatory | Description |
|---|---|---|---|
| spec | {weak} | no | permits reassignment of activities |

**Table 6.11: Parameters for *Binding of Duty***

#### Example

$$\ll \textit{BoD: } spec = \text{“}weak\text{”} \gg$$

The BoD annotations in Figure 6.1 mean that an individual learner cannot pass responsibility to carry out activities to other role holders of learner, while a re-assignment of activities of placement coordinators is possible.

#### Transformation

| BP Security Policy | Process Adaption | Parameters for BP security components |
|---|---|---|
| x | | x |

## Delegation

#### Description

Delegating access rights may apply to BPMN activities, so-called *task-based delegation*, but also to data objects. To differentiate between these two kinds of delegations, we introduce ≪T-Delegation≫ and ≪D-Delegation≫. We now describe data-based delegation. Process participants can delegate rights to access data objects to someone who currently does not have the authorization. In our context, delegation deals with access rights to data which is external or is controlled by the same process; see the data handling in BPMN 2.0 (Section 10.3 in [44]). Delegation varies depending on the category of the data concerned and therefore requires different parameters. We see two categories: A *data object* is local to the process, and its lifetime is bound to the process where it is defined; *data stores* are external data sources which persist beyond the lifetime of a process. For instance, this could be a database, say, with personal data, like e-health or e-portfolio data. For internal and external data, there can exist access restrictions. Rules to access *data stores* are often specified using external mechanisms. Access rights to *data objects* in turn have to be defined explicitly within the BP. During process execution, role holders who have the right to

access the data in question might lose it. Next, external services called need access to external data which a role holder but not the service itself is authorized to. These are situations requiring delegation of access rights.

#### Syntax

*≪ D-Delegation: rights="list ($rightsname)" object="$objectname" interval="($activityname1,$activityname2)|$group" owner="$rolename" receiver="$rolename"|"$webservicename" spec="$specification" ≫*

| Annotation | Activities | Gr. of Activ. | Pools&Lanes | Data | Messageflows | Events |
|---|---|---|---|---|---|---|
| D-Delegation | X | X | X | X | X | - |
| T-Delegation | X | X | X | - | - | - |

**Table 6.12: Allowed BPMN elements for Delegation**

| Attribute name | Value facet | Obligatory | Description |
|---|---|---|---|
| rights | r ⊆{read, write, delete}\|$policy | no | delegated access rights for data |
| object | String | yes | delegation target |
| interval | String \| [String, String] | no | process interval the delegation is valid for |
| owner | String | context specific | owner is a roleholder of the rolename specified |
| receiver | String | context specific | receiver of the delegated rights is a roleholder, or a ws-name |
| spec | {grant, transfer} | no | also passes the right for further delegation, "transfer" additionally withdraws the rights from the owner |

**Table 6.13: Parameters for Delegation**

Further explanations of attributes:
The delegation of access rights to data has $read$, $write$, $delete$, or $policy$ as possible values for rights. If there is a delegation of access rights which are attached to the data object in form of a sticky policy, the $rights$ parameter is optional, but in this case the $object$ parameter is required. The parameters $owner$ and $spec$ are optional. The specified rights or the rights of the object specified in the policy are delegated. The rights describe which use of the parameter $\$objectname$ is allowed. The validity of a delegation can be specified for the execution time of activities in the interval [$\$activityname1$, $\$activityname2$], i.e., the part of the process starting with $\$activityname1$ and ending with $\$activityname2$, or in the $\$group$ of activities. The owner before delegation is specified using either a $rolename$ explicitely given or implicitly defined by a $poolname$ or $lanename$, which are used as rolenames. The same holds for the receiver. In addition, the receiver might be a $webservicename$. $spec$ denotes a delegation with $grant$ rights for the delegate. Annotating data elements of BPMN is a current restriction of the graphical modelling tool used, but it should be allowed (as soon as it is technically possible).

#### Example

Figure 6.2 shows a process activity dealing with a personal data store (PDS). The annotation ≪ D-Delegation ... ≫ gives access rights to the "$View-PDS-of-Learner" from the holder of role "Placement coordinator" to the web service of type "PDS".

The annotation

*≪ T-Delegation: rights="$rightsname" object="(activity, $activityname")|"(interval,*

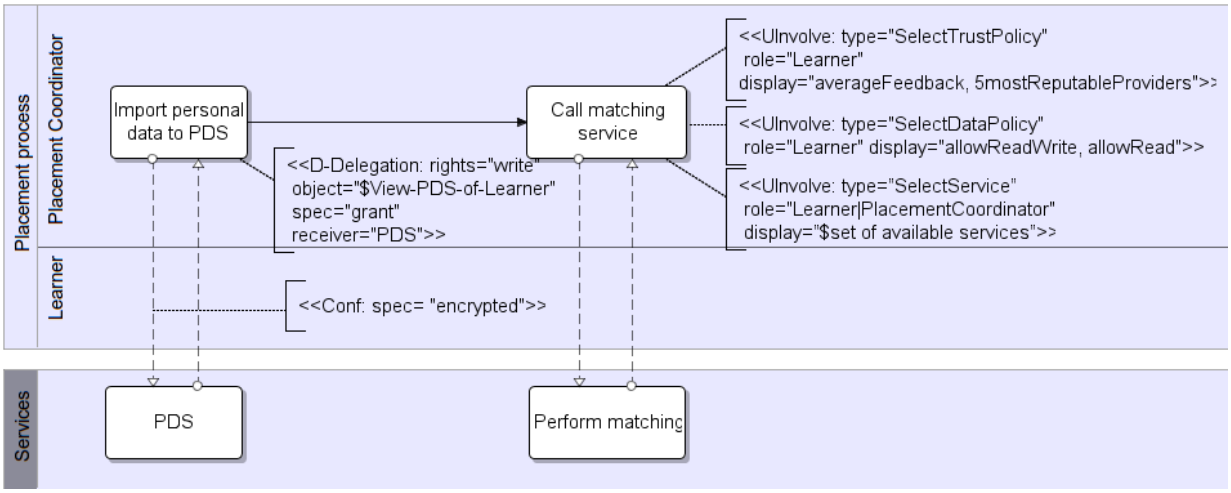


**Figure 6.2: Example Security Annotations for Process Activity**

*($activityname1,$activityname2))|"(group,$group)"|"(pool, $poolname)"|"(lane, $lanename)" owner="$rolename" receiver="$rolename" spec="$specification" ≫*

specifies the delegation of access rights to tasks, with $execute$ as possible value for rights. Because only one value is possible, this parameter is optional for T-Delegation. The tasks are defined by the $object$ parameter, which can be an activity, the activities within the interval [$\$activityname1$, $\$activityname2$], i.e., the part of the process starting with $\$activityname1$ and ending with $\$activityname2$, or a $\$group$ of activities, a pool or a lane. The owner of the rights before delegation is specified by the parameter $owner$ and is either a $rolename$ explicitely given or implicitly defined by a $poolname$ or a $lanename$, which are assigned as rolenames. $spec$ denotes a delegation with $grant$ rights for the delegate. The parameters $object$, $owner$ and $spec$ are optional. Either $object$ or $owner$ must be defined, the other not speciefied parameter is the annotated BPMN element.

#### Transformation

| BP Security Policy | Process Adaption | Parameters for BP security components |
|---|---|---|
| x | | |

## Adaptation

#### Description

To give way to ad-hoc process changes, but in a controlled way, we allow the specification of rights to adapt process instances. (Annotation **adaptation** in Table 6.1). With

$$\ll \textit{Adaptation: } rights = \text{“}\$rightsname\text{”}\ role = \text{“}\$rolename\text{”} \gg$$

we give rights to adapt (i.e., to migrate) process instances. By means of the parameters $role =$ “$\$rolename$” and $rights =$ “$\$rightsname$” (the predefined vocabulary for rights is $insert$, $update$, $delete$) we specify which role holders have particular rights to adapt the process at the annotated position. Consequently, role holders with the appropriate rights can trigger and migrate running process instances. For example, a student in an e-learning scenario wants to see who accessed his personal data, although the process does not support this activity. Or a coach might decide that a student has to pass further exams, which are not foreseen yet. With the rights to adapt (migrate) the process, both users can adapt the process instance according to their needs. Adaptation rights are defined for any BPMN elements (activities, group of activities, pools and lanes, data, message flows, and events).

From the realization point of view, these activities are managed as abstract web services and are maintained typically as process fragments in particular repositories. When users want to plug-in non-foreseen

process parts (single activities or sub-processes) into the process, they can select the appropriate process fragments accordingly.

In our approach, we additionally support a second way for process adaptations. The annotation of user involvements dealing with security and trust issues lead to modifications of the process model. We describe this in Section 6.5 and Section 4.6.4).

#### Syntax

$$\ll \textit{Adaptation: } rights = \text{“}\$rightsname\text{”}\ role = \text{“}\$rolename\text{”}\ pre = \text{“}\$precondition\text{”}$$
$$post = \text{“}\$postcondition\text{”} \gg$$

| Annotation | Activities | Gr. of Activ. | Pools&Lanes | Data | Messageflows | Events |
|---|---|---|---|---|---|---|
| Adaptation | X | X | X | X | X | X |

**Table 6.14: Allowed BPMN elements for *Adaptation***

| Attribute name | Value facet | Obligatory | Description |
|---|---|---|---|
| rights | {insert, update, delete} | yes | allows to detail how to adapt the process |
| role | String | yes | |

**Table 6.15: Parameters for *Adaptation***

#### Transformation

| BP Security Policy | Process Adaption | Parameters for BP security components |
|---|---|---|
| X | | |

Table 6.16 summarizes the allowed authorization constraints.

| Annotation Term | Roles | Activ. | Gr. of Activ. | Pools&Lanes | Data | Messagef. | Events |
|---|---|---|---|---|---|---|---|
| Role Assignment | - | X | X | X | - | - | - |
| Assignment Mechanism | X | X | X | X | - | - | - |
| User Assignment | - | X | X | X | - | - | - |
| Binding of Duty | - | X | X | X | - | - | - |
| Separation of Duty | - | X | X | X | - | - | - |
| D-Delegation | - | X | X | X | - | X | - |
| T-Delegation | - | X | X | X | - | X | - |
| Adaptation | - | X | X | X | X | X | X |

**Table 6.16: Allowed Authorization Annotations**

## 6.2 Authentication

## Authentication

#### Description

The annotation $\ll$ *Authn: attribute*= $list(\$attributename, \$value)$
$idp = \text{“}\$identityprovider\text{”} \gg$
means that process participants must be authenticated and identified. The task of an identity provider (IdP)

is to certify attributes of process participants. This certification leads to authentications. For this purpose, we specify a list of attributes which a particular identity provider (IdP) can certify.

#### Syntax

$\ll$*Authn: attribute=*$list(\$attributename, \$value)$ $idp =$ "$\$identityprovider$"$\gg$

| Annotation | Activities | Gr. of Activ. | Pools&Lanes | Data | Messageflows | Events |
|---|---|---|---|---|---|---|
| Authentication | X[1] | [2] | [3] | - | - | - |
| Auditing | X | X | X | X | X | X |
| Confidentiality | - | - | - | - | X | - |

**Table 6.17: Allowed BPMN elements for *Authentication***

| Attribute name | Value facet | Obligatory | Description |
|---|---|---|---|
| attribute | String | yes | |
| idp | String | yes | IdP to be used for authentication |

**Table 6.18: Parameters for *Authentication***

#### Example

For instance, any holder of role "learner" in Figure 6.1 must be authenticated as a student at the University of Nottingham and identified by an identity provider given or a related one. The term $\ll$Authn$\gg$ can be annotated for pools and lanes (which will be interpreted as roles).

#### Transformation

| BP Security Policy | Process Adaption | Parameters for BP security components |
|---|---|---|
| x | | |

## 6.3 Auditing

### Audit

#### Description

The annotation $\ll$*Audit:* $policy =$ "$\$policyname$"$\gg$
enforces a monitoring of the execution of an activity, a group of activities, data access, event, or message flow.

Various legal requirements call for different logging activities. For example, if a process handles personal data, it must be logged, among others, who performed which action upon which data, and at which time [45]. Thus, we specify the objects to be logged in an auditing policy (parameter $policy =$ "$\$policyname$"). To illustrate, we have annotated the process "start event" with the annotation term $Audit$ in Figure 6.1. This specifies a monitoring of consenting to terms and conditions at the process start according to the policy "auditAll".

#### Syntax

$\ll$*Audit:* $policy =''$ $\$policyname$"$\gg$

| Annotation | Activities | Gr. of Activ. | Pools&Lanes | Data | Messageflows | Events |
|---|---|---|---|---|---|---|
| Auditing | X | X | X | X | X | X |

**Table 6.19: Allowed BPMN elements for *Auditing***

| Attribute name | Value facet | Obligatory | Description |
|---|---|---|---|
| policy | String | yes | audit policy |

**Table 6.20: Parameters for *Auditing***

#### Example

$\ll$*Audit: policy =*$''$ $TAS3LoggingPolicy$"$\gg$

#### Transformation

| BP Security Policy | Process Adaption | Parameters for BP security components |
|---|---|---|
| | x | x |

## 6.4 Data and Message Flow Security

### Confidentiality

#### Description

Authorization mechanisms support a basic level of confidentiality, because only authorized humans or web services may perform activities and access data. To improve confidentiality further, the system can also protect message flows. This can be expressed by:

$\ll$*Conf: spec=*"*$value*"$\gg$

We allow the parameter "spec". Its possible values are "$encrypted$" or "$signed$". For example, the data flow from activity "Import personal data to PDS" to activity "PDS" in Figure 6.2 must be encrypted.

#### Syntax

$\ll$ *Conf: spec =*$''$ $\$value$" $\gg$

| Annotation | Activities | Gr. of Activ. | Pools&Lanes | Data | Messageflows | Events |
|---|---|---|---|---|---|---|
| Confidentiality | - | - | - | - | X | - |

**Table 6.21: Allowed BPMN elements for *Confidentiality***

#### Example

$\ll$ *Conf: spec =*$''$ $encrypted, signed$" $\gg$

#### Transformation

| BP Security Policy | Process Adaption | Parameters for BP security components |
|---|---|---|
| | | x |

Table 6.23 summarizes the usage of these annotation terms.

| Attribute name | Value facet | Obligatory | Description |
|---|---|---|---|
| spec | {encrypted, signed} | yes | Message is sent encypted and/or signed |

**Table 6.22: Parameters for *Confidentiality***

| Annotation | Activities | Gr. of Activ. | Pools&Lanes | Data | Messageflows | Events |
|---|---|---|---|---|---|---|
| Authentication | X[4] | [5] | [6] | - | - | - |
| Auditing | X | X | X | X | X | X |
| Confidentiality | - | - | - | - | X | - |

**Table 6.23: Allowed BPMN elements for other Security Goals**

## 6.5 User Involvements

By studying real-world applications that handle personal data in particular, we have identified user involvements that are specific in the security and trust context. These user involvements are important, because users have to specify and agree at runtime to security preferences that are context-specific. Having examined various application scenarios, we have identified the following security- and trust-specific user involvements: giving consent, selecting services according to required trust levels, specifying data access policies or trust levels, specifying users' interaction preferences (i.e., their preferred degree of direct involvement in security issues), and trust-feedback mechanisms. To support the representation of such involvements, we introduce annotation terms to specify them declaratively. Thus, the process designer simply annotates user involvements instead of modelling them explicitly. We now describe the annotation terms that represent security-specific user involvements.

The representation is as follows:

$$\ll UInvolve{:}list(\$parametername{=}\$value) \gg$$

Each user involvement has a parameter of name *type*, and its value specifies the intention of a user, e.g., *type* = *Consent* means giving user consent. Table 6.24 lists our language for security- and trust-specific user involvements. Dots indicate further parameters, such as specification of users' interaction preferences, giving user consent or trust feedback, service selection, specification of a data policy, trust policy setting or selection. We can annotate activities and some events of a process model with user involvements. Unless specified otherwise, these involvements are invoked before the annotated activity takes place. To deal with other cases, there is a parameter $insertplace$, indicating the position of the involvement in the flow of the process. The interaction is inserted *before* the specified activity.

| User Involvement | Syntax |
|---|---|
| Set Interaction Preferences | ≪UInvolve: type="SetIAPref" ... ≫ |
| Give Consent | ≪UInvolve: type="Consent" ... ≫ |
| Service Selection | ≪UInvolve: type="SelectService" ... ≫ |
| Set Data Access Policy | ≪UInvolve: type="SetDataPolicy" ... ≫ |
| Select Data Access Policy | ≪UInvolve: type="SelectDataPolicy" ... ≫ |
| Set Trust Policy | ≪UInvolve: type="SetTrustPolicy" ... ≫ |
| Select Trust Policy | ≪UInvolve: type="SelectTrustPolicy" ... ≫ |
| Give Trust Feedback | ≪UInvolve: type="TrustFeedback" ... ≫ |

**Table 6.24: Overview of User Involvements**

## Set Interaction Preferences

#### Description

The annotation of the first line in Table 6.24 specifies the interaction preferences of a user for a business process. These preferences may concern, for example, data access, or preferred interaction behavior concerning web-service selection.

With the annotation

$$\ll UInvolve:\ type = \text{“}SetIAPref\text{”}\ display = \text{“}list(\$option)\text{”}\ role = \text{“}\$rolename\text{”}\ insertplace = \text{“}\$activityname\text{”}\gg$$

a user can specify its preferences with respect to interactions for a business process. These preferences may concern, for example, data access, or its preferred interaction behavior concerning web service selection. The parameters $role$ and $insertplace$ are optional.

#### Syntax

$$\ll UInvolve:\ type =\!\!''\ SetIAPref\text{”}\ display =\!\!''\ list(\$option)\text{”}\ role =\!\!''\ \$rolename\text{”}\ insertplace =\!\!''\ \$activityname\text{”}\gg$$

| User Involvement | Activities | Gr. of Activ. | Pools&Lanes | Data | Messageflows | Events |
|---|---|---|---|---|---|---|
| Set Interact. Preferences | X | X | - | - | - | X |

**Table 6.25: Allowed BPMN elements for *Set Interaction Preferences***

| Attribute name | Value facet | Obligatory | Description |
|---|---|---|---|
| display | String | yes | list of options presented to user |
| role | String | no | role holder the interaction is assigned to |
| insertplace | String | no | place in the process where the interaction gets inserted |

**Table 6.26: Parameters for *Set Interaction Preferences***

#### Example

The annotation is illustrated by the following example:

$\ll UInvolve:\ type = \text{“}SetIAPref\text{”}\ display =$“web service selection: ask anytime, do not ask”.$\gg$

A user selects his interaction preferences from the options “ask anytime” or “do not ask” in case that security issues occur during process execution.

#### Transformation

| BP Security Policy | Process Adaption | Parameters for BP security components |
|---|---|---|
| | x | |

## User Consent

#### Description

Due to legal regulations, user consent for privacy terms is needed when personal data is managed (Table 6.24). Process designers have this domain knowledge and can represent it accordingly.

The annotation

$$\ll UInvolve:\ type =\!\!''\ Consent''\ ...\gg$$

specifies that a user must give consent to some privacy terms, otherwise the process ends. The $target$ identifies the particular privacy terms that must be accepted by a user (type string), $display$ specifies the displayed text options of a user interface a user can choose for agreement.

#### Syntax

$\ll UInvolve:\ type =\!\!''\ Consent"\ display =\!\!''\ list(\$option)"\ target =\!\!''$ \$targetname of privacy terms" $role =\!\!''$ \$rolename" $insertplace =\!\!''$ \$activityname" $\gg$

| User Involvement | Activities | Gr. of Activ. | Pools&Lanes | Data | Messageflows | Events |
|---|---|---|---|---|---|---|
| User Consent | X | X | - | - | - | X |

**Table 6.27: Allowed BPMN elements for *User Consent***

| Attribute name | Value facet | Obligatory | Description |
|---|---|---|---|
| display | String | yes | list of options presented to user |
| target | String | yes | terms and conditions for which consent is need |
| role | String | no | role holder the interaction is assigned to |
| insertplace | String | no | place in the process where the interaction gets inserted |

**Table 6.28: Parameters for *User Consent***

#### Example

For illustration, we give a brief example where a user has to choose between "yes" and "no" of the $TAS^3$ Privacy Terms:

$$\ll UInvolve:\ type = \text{“}Consent\text{”}\ display = \text{“}yes, no\text{”}\ target = \text{“}TAS3PrivacyTerms\text{”}\gg$$

Figure 6.1 shows two annotated user involvements aiming to consent to the placement process as well as to privacy terms when the process starts.

#### Transformation

| BP Security Policy | Process Adaption | Parameters for BP security components |
|---|---|---|
| | x | |

## Service Selection

#### Description

The term

$$\ll UInvolve:\ type = \text{“}SelectService\text{”}\ display = \text{“}list(\$option)\text{”}\ role = \text{“}\$rolename\text{”}$$
$$insertplace = \text{“}\$activityname\text{”}\gg$$

means that the user who performs the annotated activity should first be asked to select a service with a specific trust level from the set of available services. A service-discovery mechanism using trust characteristics of services obtains the available services in advance. The result is shown to the user, in line with the variable $display = \text{“}list(\$option)\text{”}$. Optionally, the process modeller can specify a role name, stating that a holder of the role has to select the service. A prerequisite for service discovery is that a user has specified his trust policy (see below).

#### Syntax

$$\ll UInvolve:\ type =''\ SelectService''\ display =''\ list(\$option)''\ role =''\ \$rolename''\ insertplace =''\ \$activityname''\ \gg$$

| User Involvement | Activities | Gr. of Activ. | Pools&Lanes | Data | Messageflows | Events |
|---|---|---|---|---|---|---|
| Service Selection | X | X | - | - | - | - |

**Table 6.29: Allowed BPMN elements for *Service Selection***

| Attribute name | Value facet | Obligatory | Description |
|---|---|---|---|
| display | String | yes | List of services to be selected |
| role | String | no | role holder the interaction is assigned to |
| insertplace | String | no | place in the process where the interaction gets inserted |

**Table 6.30: Parameters for *Service Selection***

#### Example

$$\ll UInvolve:\ type =''\ SelectService''\ display =''\ \$setOfAvailableServices''\ role =''\ Learner''\ insertplace =''\ Callmatchingservice''\ \gg$$

Figure 6.2 contains three user involvements, one representing a service selection. A learner or a placement coordinator has to select one of the services available.

#### Prerequisites

Preceding *Set Trust Policy* or *Select Trust Policy* annotation:
A trust policy must be specified. This means a *Service Selection* annotation requires an additional *Set Trust Policy* or *Select Trust Policy* annotation being attached to a task *preceding* the task which is annotated with *Service Selection*.

#### Transformation

| BP Security Policy | Process Adaption | Parameters for BP security components |
|---|---|---|
| | x | |

## Data-Access Policy

When a process manages sensitive data, the data owner can select security policies for this data. This can be specified either by user involvements of type “Set Data Policy” or of type ”Select Data Policy”,

## Set Data Access Policy

#### Description

The annotation

$$\ll UInvolve:\ type = \text{“}SetDataPolicy\text{”}\ policy = \text{“}\$policyname\text{”}\ target = \text{“}\$namesensitivedata\text{”}\ role = \text{“}\$rolename\text{”}\ insertplace = \text{“}\$activityname\text{”}\gg$$

sets a data policy. The parameters $role$ and $insertplace$ are optional.

#### Syntax

$$\ll UInvolve:\ type = \text{“}SetDataPolicy\text{”}\ policy = \text{“}\$policyname\text{”}\ target = \text{“}\$namesensitivedata\text{”}\ role = \text{“}\$rolename\text{”}\ insertplace = \text{“}\$activityname\text{”} \gg$$

| User Involvement | Activities | Gr. of Activ. | Pools&Lanes | Data | Messageflows | Events |
|---|---|---|---|---|---|---|
| Set Data Access Policy | X | X | - | - | - | X |

**Table 6.31: Allowed BPMN elements for *Set Data Access Policy***

| Attribute name | Value facet | Obligatory | Description |
|---|---|---|---|
| policy | String | yes | data policy to be set |
| target | String | yes | data for which the policy is set |
| role | String | no | role holder the interaction is assigned to |
| insertplace | String | no | place in the process where the interaction gets inserted |

**Table 6.32: Parameters for *Set Data Access Policy***

#### Example

$$\ll UInvolve:\ type = \text{“}SetDataPolicy\text{”}\ target = \text{“}e - portfoliodata\text{”}\ policy = \text{“}allowAll\text{”} \gg$$

gives the rights for e-portfolio data of a user.

#### Transformation

| BP Security Policy | Process Adaption | Parameters for BP security components |
|---|---|---|
| | x | |

## Select Data Access Policy

#### Description

The annotation term

$$\ll UInvolve:\ type = \text{“}SelectDataPolicy\text{”}\ target = \text{“}\$nameofsensitivedata\text{”}\ display = \text{“}list(\$option)\text{”}\ role = \text{“}\$rolename\text{”}\ insertplace = \text{“}\$activityname\text{”} \gg$$

enables that a user specifies a security policy for access to his sensitive data by selecting from a policy list. As parameters, we have target=“name of sensitive data” (type string) and display=“option1, option2,..” (of type list of strings).

#### Syntax

$$\ll UInvolve:\ type = \text{“}SelectDataPolicy\text{”}\ target = \text{“}\$nameofsensitivedata\text{”}\ display = \text{“}list(\$option)\text{”}\ role = \text{“}\$rolename\text{”}\ insertplace = \text{“}\$activityname\text{”} \gg$$

#### Example

The following example illustrates the annotation:

$$\ll UInvolve:\ type = \text{“}SelectDataPolicy\text{”}\ target = \text{“}e - portfoliodata\text{”}\ display = \text{“}denyAll, allowRead, allowAll\text{”}\ role = \text{“}\$rolename\text{”}\ insertplace = \text{“}\$activityname\text{”} \gg$$

| User Involvement | Activities | Gr. of Activ. | Pools&Lanes | Data | Messageflows | Events |
|---|---|---|---|---|---|---|
| Select Data Access Policy | X | X | - | - | - | X |

**Table 6.33: Allowed BPMN elements for *Select Data Access Policy***

| Attribute name | Value facet | Obligatory | Description |
|---|---|---|---|
| target | String | yes | data for which the policy is selected |
| display | String | yes | policies available for selection |
| role | String | no | role holder the interaction is assigned to |
| insertplace | String | no | place in the process where the interaction gets inserted |

**Table 6.34: Parameters for *Select Data Access Policy***

The user can decide whether all process participants are denied or allowed to access its e-Portfolio data. Process modeller can also consider the case that data policies might be too restrictive. In this case, potential process adaptation by means of a modification of data policies can be modelled.

#### Transformation

| BP Security Policy | Process Adaption | Parameters for BP security components |
|---|---|---|
| | x | |

## Trust Policies

We characterize service providers by trust levels, representing their reputation gained in the past. Process participants can specify the minimum trust level of service providers by means of a trust policy. Users can either set (i.e., specify) or select trust policies interactively from a list (see lines 6 and 7 in Table 6.24).

## Set Trust Policy

#### Description

A trust policy will be set (i.e., initialized or modified) in a process instance by the annotation

$$\ll \textit{UInvolve: } type = \text{“}SetTrustPolicy\text{”}\ policy = \text{“\$}policyname\text{”}\ role = \text{“\$}rolename\text{”}\ insertplace = \text{“\$}activityname\text{”} \gg$$

. The parameter $name$ identifies the trust policy to be used.

#### Syntax

$$\ll \textit{UInvolve: } type = \text{“}SetTrustPolicy\text{”}\ policy = \text{“\$}policyname\text{”}\ role = \text{“\$}rolename\text{”}\ insertplace = \text{“\$}activityname\text{”} \gg$$

| User Involvement | Activities | Gr. of Activ. | Pools&Lanes | Data | Messageflows | Events |
|---|---|---|---|---|---|---|
| Set Trust Policy | X | X | - | - | - | X |

**Table 6.35: Allowed BPMN elements for *Set Trust Policy***

#### Example

The following example illustrates the setting of the trust policy “trustAll”:

$$\ll \textit{UInvolve: } type = \text{“}SetTrustPolicy\text{”}\ policy = \text{“}trustAll\text{”} \gg$$

| Attribute name | Value facet | Obligatory | Description |
|---|---|---|---|
| policy | String | yes | trust policy to use |
| role | String | no | role holder the interaction is assigned to |
| insertplace | String | no | place in the process where the interaction gets inserted |

**Table 6.36: Parameters for *Set Trust Policy***

#### Transformation

| BP Security Policy | Process Adaption | Parameters for BP security components |
|---|---|---|
| | x | |

## Select Trust Policy

#### Description

The following annotation specifies the selection of a trust policy:

$$\ll UInvolve:\ type = \text{“}SelectTrustPolicy\text{”}\ display = \text{“}list(\$option)\text{”}\ role = \text{“}\$rolename\text{”}\ insertplace = \text{“}\$activityname\text{”}\gg$$

#### Syntax

$$\ll\ UInvolve:\ type = \text{“}SelectTrustPolicy\text{”}\ display = \text{“}list(\$option)\text{”}\ role = \text{“}\$rolename\text{”}\ insertplace = \text{“}\$activityname\text{”}\ \gg$$

| User Involvement | Activities | Gr. of Activ. | Pools&Lanes | Data | Messageflows | Events |
|---|---|---|---|---|---|---|
| Select Trust Policy | X | X | - | - | - | X |

**Table 6.37: Allowed BPMN elements for *Select Trust Policy***

| Attribute name | Value facet | Obligatory | Description |
|---|---|---|---|
| display | String | yes | Trust policies available for selection |
| role | String | no | role holder the interaction is assigned to |
| insertplace | String | no | place in the process where the interaction gets inserted |

**Table 6.38: Parameters for *Select Trust Policy***

#### Example

For example, the following options can be selected by a user:

$$\ll UInvolve:\ type = \text{“}SelectTrustPolicy\text{”}$$
$$display = \text{“}trustAll, averageFeedback, mostreputableUsers, 5mostreputableUsers\text{”}\gg$$

The user involvement of type “SelectTrustPolicy” in Figure 6.2 means that a learner has to select a trust policy from the available options.

#### Transformation

| BP Security Policy | Process Adaption | Parameters for BP security components |
|---|---|---|
| | x | |

## Give Trust Feedback

#### Description

The annotation

$$\ll UInvolve:\ type = \text{“}TrustFeedback\text{”}\ display = \text{“}list(option)\text{”}\ role = \text{“}\$rolename\text{”}\ insertplace = \text{“}\$activityname\text{”}\gg$$

means that users who hold the role calling the web service are asked for feedback about its trustworthiness. Here, we use the optional parameter $insertplace$. It specifies where exactly in the process model users give feedback. The parameter role is optional.

#### Syntax

$$\ll UInvolve:\ type = \text{“}TrustFeedback\text{”}\ display = \text{“}list(option)\text{”}\ role = \text{“}\$rolename\text{”}\ insertplace = \text{“}\$activityname\text{”}\gg$$

| User Involvement | Activities | Gr. of Activ. | Pools&Lanes | Data | Messageflows | Events |
|---|---|---|---|---|---|---|
| Give Trust Feedback | X | X | - | - | - | - |

**Table 6.39: Allowed BPMN elements for *Give Trust Feedback***

| Attribute name | Value facet | Obligatory | Description |
|---|---|---|---|
| display | String | yes | available feedback options |
| role | String | no | role holder the interaction is assigned to |
| insertplace | String | no | place in the process where the interaction gets inserted |

**Table 6.40: Parameters for *GiveTrustFeedback***

#### Example

The annotation

$$\ll UInvolve:\ type = \text{“}TrustFeedback\text{”}\ display = \text{“}positive, neutral, negative, nofeedback\text{”}\ insertplace = \text{“}Endprocess\text{”}\gg$$

means that users can pick a feedback value from the options listed. This user involvement will be triggered before activity *End process* executes.

#### Transformation

| BP Security Policy | Process Adaption | Parameters for BP security components |
|---|---|---|
| | x | |

Table 6.41 summarizes how user involvements can be annotated in BPMN diagrams.

In summary, our language represents security aspects and security- and trust-specific user involvements. Having taken into account all requirements from two complex, realistic application scenarios, we argue that our language is the result of a more comprehensive design effort than previous ones.

| User Involvement | Activities | Gr. of Activ. | Pools&Lanes | Data | Messageflows | Events |
|---|---|---|---|---|---|---|
| Set Interact. Preferences | X | X | - | - | - | X |
| Give Consent | X | X | - | - | - | X |
| Service Selection | X | X | - | - | - | - |
| Set Data Access Policy | X | X | - | - | - | X |
| Select Data Access Policy | X | X | - | - | - | X |
| Set Trust Policy | X | X | - | - | - | X |
| Select Trust Policy | X | X | - | - | - | X |
| Give Trust Feedback | X | X | - | - | - | - |

**Table 6.41: Allowed BPMN elements for User Involvements**

## Amendment History

| Ver | Date | Author | Description/Comments |
|---|---|---|---|
| 3.4 | 03.11.2011 | Jutta | final revision |
| 3.3 | 31.10.2011 | Jutta | revise introduction, conclusions |
| 3.2 | 21.10.2011 | Jens, Jutta | limitations, policies, Annex A added, adaptation |
| 3.1 | 05.08.2011 | Thorsten | additions to BPMN elements in Annotation Language |
| 3.0 | 29.12.2010 | all | address internal reviewing comments, revise sec.modelling, adaptation |
| 2.5 | 24.12.2010 | Jutta | introduction, conclusion, transform sec.modelling, adaptation |
| 2.4 | 22.12.2010 | Silvia, Jutta, Jens | new parts in chapter 4 and 5 and revision |
| 2.3 | 18.12.2010 | Jens, Jutta | new parts in chapter 4 and 5 |
| 2.2 | 03.12.2010 | Silvia, Thorsten, Ioana | new parts in chapter 4 |
| 2.1 | 15.10.2010 | Thorsten | First draft new parts |