TAS^3 Deliverable D3.3 - 3rd iteration:
Integration with TAS^3 trust applications of employment and eHealth processes (3)

Accompanying Letter to the Reviewers

In the report of the TAS^3 review on March 11, 2011, the project reviewers had comments on Deliverable D3.3 (1) “Integration with TAS^3 trust applications of employment and eHealth processes (1)” delivered at M24. The TAS^3 Consortium has addressed the reviewer comments as follows.

a). The report does not properly justify why the analysis of the eHealth scenario is not included.

We have now addressed this issue in Section 6. This is a new section providing an evaluation of our security components in three pilot demonstrators. In particular, see the concluding Section 6.3 for a discussion.

b) The Nottingham use case of Student Placements is the only case described, and integration/implementation challenges are poorly described.

The first iteration of D3.3, as of M24, describes the realization of the Student Placements application with a business process. It is operational on the business process management system and does support security and trust constraints. The goal of this first demonstrator has been to come up with a first integration of the security components of the TAS3 security framework that have already been available at that time. Back then, having another use case in parallel, with the TAS3 security framework not yet available in full beauty, would not have given us significantly deeper insights regarding the integration. Following this milestone, we have however applied the advanced business-process-security components to further demonstrators. Additionally, we have elaborated a BTG use case in the e-health domain, see Comment a). The description of the implementation of the demonstrator applications has been work of the subsequent project period and has been described in the second iteration of the deliverable, due in M36. Please see again Sections 2.3 and 2.4. Section 6 has been introduced as new in the final version of the deliverable and describes challenges and experiences stemming from the implementation and deployment of our security components and the integration of further functionality and components in the business processes and the BPMS.

c) The modeling of “break the glass” and of obligations is missing, as is the inclusion of business rules.

The new Section 5, namely “Break-the-Glass Functionality for Business Processes”, describes how to integrate BTG functionality into business processes and provides a discussion. Obligations are an essential part of a BTG scenario and are discussed in the same section as well. Please observe that modelling of 'break the glass' and of

obligations has not been part of the description of work regarding WP3, but we have covered it nevertheless.
Our approach realizes BTG in BPs. It supports all levels from modeling to execution in our secure BP-engine. At the modeling level, we have extended our security-annotation language with BTG-specific annotations. We have applied the BTG functionality to an e-health use case, dealing with access to patient data.
Regarding business rules, the description of work for WP3 does not include this topic. Business rules are one out of several ways to specify security characteristics of business processes. However, they are limited (we argue), as they do not allow to deal with security-relevant user interactions and adaptations of the process structure in an elegant way, to give examples. Our annotation language in turn facilitates this. It also is intuitively understandable, as user experiments we have recently conducted have shown.

d) The flexibility of the approach is unclear.

Section 6 summarizes our experiences from the deployment of our security components in the use case applications and in the TAS3 security framework. The following items are important with respect to flexibility as well:

- We have deployed our security-annotation language in three demonstrators and in the BTG e-health scenario. This shows that our approach is flexible in that it supports different scenarios.
- Our approach features user involvements, i.e., lets individuals specify the trust policies of their choice, set preferences how tightly to be involved in security-relevant decisions, e.g., in selecting services. Here, flexibility means that the extent of taking user preferences into account and of letting users influence security and trust parameters during the process execution is very high.
- Our annotation language allows to configure security constraints. For instance, one can specify which IdP should be used in a BP, how the context of the BP service provider can be taken into account, if applicable.

In D3.2, Section 3 describes the overall architecture of our system. Its main advantage is the synthesis of modeling security and enforcing it with a secure BPMS in a security framework like TAS3. This shows a further level of flexibility in configuring the BPMS for certain business process executions.

Next, we have evaluated the security-annotation language in user experiments with a large user base. The results provide evidence of the usability of the language for individuals who are familiar with modeling but not necessarily with security specification. For further details we refer to Section 6.4 of the deliverable.

The WP03 Team

SEVENTH FRAMEWORK PROGRAMME
Challenge 1
Information and Communication Technologies

Trusted Architecture for **Securely Shared Services**

Document Type:	Deliverable
Title:	**Integration with TAS^3 trust applications of employment and eHealth processes**
Work Package:	WP3
Deliverable Nr:	D3.3, third iteration
Editor:	Jutta Mülle, KIT – Karlsruhe Institute of Technology
Dissemination:	PU
Preparation Date:	October 16, 2011
Version:	3.0

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	KARL
2	Jens Müller	KARL
3	Thorsten Haberecht	KARL
4	Silvia von Stackelberg	KARL
5	Matthias Bracht	KARL
6	Andreas Pashaldis (Reviewer)	KUL
7	Michael Kutscher (Reviewer)	KOLD

Executive Summary

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

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

The topic of work package 3 is the support of adaptive, secure business processes in the TAS^3 architecture.

This document has the purpose to describe the application of secure business processes to use cases of the application domains of employability and e-health and to validate the results of the conceptual design and of the mechanisms of secure business processes. Deliverable D3.1 ([1]) contains the conceptual design and Deliverable D3.2 ([2]) the implementation of the components of the respective version.

We started with a use case of the employability domain, the Nottingham student placement scenario and developed together with other partners an integrated prototype implementation, as described in Deliverable D9.1.2 ([3]). In this report we first describe which of the already available components of the TAS^3 framework are used for supporting the student placement application and how it already meets the business process security requirements and what are gaps. Addressing these evaluation results we continued to develop and implement security mechanisms for business processes resulting in new iterations of Deliverable D3.1.3 in December 2010 ([1] and the next implementation status described in Deliverable D3.2.2 of June 2010 ([4]).

With the enhanced implementation of our components we have cooperated in several workshops with other partners, especially the pilot partners of WP9, on two pilot demonstrators: An enhanced second version of the Nottingham student placement scenario and the Kenteq Mass Layoff Scenario. We have evaluated these latter two demonstrators with respect to the WP3 components. Further, we have revised the description of modelling and implementation of these pilots with the final status of the demonstrators. As new application of secure BPM we have developed a use case dealing with break the glass (BTG) functionality. To this end, we started with a systematic analysis of BTG functionality to be applied in business process scenarios for TAS^3 applications. For this, two scenarios, one of the e-health domain, the other of the employability domain have been used to investigate the advantages of BP context in a BTG scenario. Next, we investigated BTG support from security annotation modelling, on the transformation to the enforcement phase and on the execution level of secure business process management. We then applied this approach to an example of the e-health domain.

The new Section 6 now describes the evaluation results of all demonstrators reported on in this deliverable. We validated how suitable and applicable the security components specific to business process management have been used in the demonstrators. This comprises the components developed for secure business process management, i.e., modelling support of security aspects, configuration of secure business process management, and security components enhancing business process management systems. We also regarded the integration of the BP-specific security components with the TAS^3 framework. For validating our security modelling approach for business processes, we have carried out user experiments for a large group of participants in Subsection 6.4.

Contents

List of Figures

1 Introduction

1.1 Scope and objectives

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 business processes
- support the security requirements of business process 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.

In the TN of the TAS^3 architecture applications can be realized as secure business processes, which provide an orchestration of web services, user interactions and use of data. In this project the focus is particularly on applications using personal identifiable information. In line with the overall direction of TAS^3, we put much emphasis on processes that handle privacy-related data and personal information. We will exemplify this by using e-portfolios used for employment and training services and health records 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.

This document describes the application of secure business processes to applications of the domains of employability and e-health. It validates the conceptual design and the mechanisms of secure business processes. Deliverable D3.1 ([5]) contains the conceptual design and Deliverable D3.2 ([6]) the implementation of the components of the respective version.

We started with a use case of the employability domain, the Nottingham student placement scenario and developed together with other partners an integrated prototype implementation, as described in Deliverable D9.1.2 ([3]). The pilot application was developed as a first demonstration during the second project year and therefore used the components available in that period. In this report we first describe which of the already available components of the TAS^3 framework are used for supporting the student placement application and how it already meets the business process security requirements and what are gaps. The first iteration of D3.3, as of M24, describes the realization of the Student Placements application with a business process. It is operational on the business process management system and does support security and trust constraints. The goal of this first demonstrator has been to come up with a first integration of the security components of the TAS3 security framework that have already been available at that time. We developed it together with other project partners, the demonstrator is described in Deliverable D9.1.2 ([3]) from the perspective of the pilots. Back then, having another use case in parallel, with the TAS3 security framework not yet available in full beauty, would not have given us significantly deeper insights regarding the integration. Following this milestone, we have however applied the advanced business-process-security components to further demonstrators. The experiences then resulted in new iterations of Deliverable D3.1.3 in December 2010 ([1]) and the next implementation status described in Deliverable D3.1.2 of June 2010 ([4]). With the enhanced implementation of our components we developed two pilot demonstrators, an enhanced second version of the Nottingham student placement scenario and the Kenteq Mass Layoff Scenario in cooperation with the pilot partners of WP9 and several technical partners.

The overview about the first Nottingham student placement demonstrator gives the high-level phases a student has to execute. As next step we describe the placement coordinator application. It is a simplified process realizing the student's personal workflow for placement. The process contains several user interactions with the student and a web service call. The service matches personal data of the student with a set of company placement offers. As personal data we use the data which are also provided by the IDP for authentication.

In order to provide a secure business process we have annotated the Business Process Modelling Notation (BPMN, [7]) diagram with security annotations. A goal of WP3 is to use BPMN annotations to produce an executable process that fulfills the security properties defined by the annotations (see [1], chapter 4.2). We addressed it in an automatic way as far as possible, but we expected remaining annotations that would need manual decisions and interventions. However, such mechanisms were not yet available at all in the period during which the first demonstrator have been implemented. Therefore, we have realized the security properties in an explicit way by enhancing the BPMN model and using calls of components of the security framework. This has been in detail described by BPMN diagrams and UML sequence diagrams.

The second Nottingham student placement demonstrator is an extension of the first one. It provides new functionality, especially refined BP security annotations and first transformation functionality. Also, it contains enhanced assignment of users to tasks, a business process policy, delegation mechanisms within the BP, accessing a personal data store (PDS) and improved user centricity. The structure of description is according to the first Nottingham student placement demonstrator.

The Kenteq Mass Layoff demonstrator allows to integrate calls to several service providers and to legacy applications. The main focus has been to demonstrate the general portal, policy management for end-users, data discovery, legacy data integration and improved dashboard and auditing functionality. We have described the effects to business processes in this report, in particular this has influenced the interaction with other components of the TAS^3 framework.

For the final report, we now have evaluated the Kenteq mass layoff demonstrator and the second student placement demonstrator, which we have demonstrated during the last reporting period. Therefore, we have also updated the annotation description of the modelling according to changes of the annotation language, and the implementation of these pilots according to the final status of the demonstrators. As new application of secure BPM we have developed a use case dealing with break the glass (BTG) functionality. This is described in Section 5. To this end, we started with a systematic analysis of BTG functionality to be applied in business process scenarios for TAS^3 applications. For this, two scenarios, one of the e-health domain, the other of the employability domain have been used to investigate the advantages of BP context in a BTG scenario. Next, we investigated BTG support from security annotation modelling, on the transformation to the enforcement phase and on the execution level of secure business process management. We then applied this approach to an example of the e-health domain.

Section 6 now summarizes the evaluation results of all demonstrators reported on in this deliverable. We validated how suitable and applicable the security components specific to business process management have been used in the demonstrators. This comprises the components developed for secure business process management. For each of these components, i.e., modelling support of security aspects, configuration of secure business process management, and security components enhancing business process management systems is described, in which way they have been used and what have been challenges in respect to integration with the application as well as with BP-specific security components with the TAS^3 framework. For validating our security modelling approach for business processes, we have carried out user experiments for a large group of participants in Subsection 6.4. The results provide evidence of the usability of the language for individuals who are familiar with modeling but not necessarily with security specification.

1.2 Document organisation

The rest of the document is organised as follows.

Section 2 describes the Nottingham placement application, a scenario that was newly built in an early phase of the project to show the integration of already available TAS^3 security and trust components. It starts with an overview in 2.1 and then describes the business process and its security annotations in Section 2.2. Then Section 2.3 gives an overview about the workflow of the whole application including the security and trust functionality and the involved components. In detail, the realization of the secure business process application is described in 2.4 along the process phases. This shows the status of integration of security components and how they achieve secure business processes. Chapter 3 and Chapter 4 describe the demonstrators developed during the third project period structured accordingly to the first demonstrator presentation. A next use case deals with BTG functionality in BPs described in Chapter 5. Chapter 6 subsumes the evaluation of all these demonstrators from the perspective of secure business-process management and the user experiments validating our security modelling annotations of business process models. The Deliverable concludes with Chapter 7.

2 The Nottingham Use Case of Student Placements – First Demonstrator

2.1 Overview

Student employability, as discussed in Deliverables D1.1 and D1.4 as well as in the previous iteration of D9.1 ([3]), is a topic that has come under increased focus in the recent economic climate. To support the expanding demands of student placements this demonstrator looks at ways in which through using services the process can be both captured and automated as much as possible. Central to this development is the exchange of sensitive and private data between all parties: University, Student, Placement provider and companies offering employment. As the demonstrator develops beyond the initial integration trials into a full pilot situation, managing trust and privacy settings between all users of the system will increase in complexity.

Efficient flow and exchange of information about the students themselves, the programmes they are eligible for and the vacancies available is needed to support the matching of learners to appropriate placements. There are elements of choice at both ends of the process: learners want to be able to choose from a selection of suitable placements, while employers wish to choose from a selection of suitable students. Preservation of anonymity and gradually staged release of data, both from the students about themselves and the placement provider (employer) about the placement help to ensure fairness and impartiality throughout the process.

In Deliverable D9.1 ([3]) the basic storyboard in section 4.2.1 describes how the application is used.

From the business process point of view, we distinguish two layers:

- the application process, and
- the technical subprocesses which handle the security support.

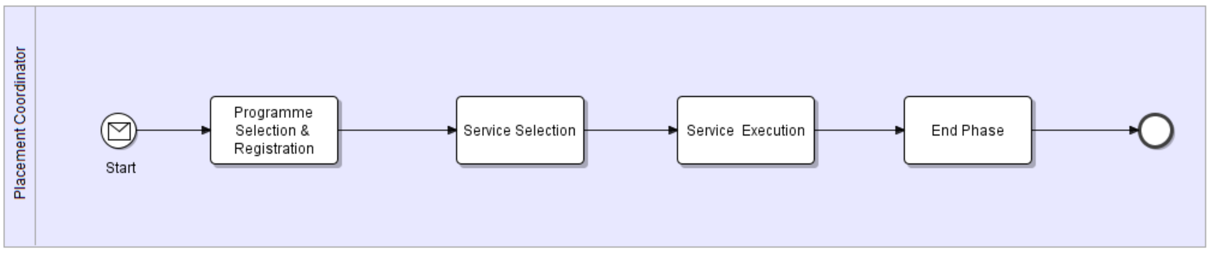


Figure 2.1: Overview of the Nottingham Student Placement Process

The pilot application was developed as a first demonstration during the second project year and therefore used the components available in that period. The application process is described in the Business Process Modelling Notation (BPMN) and was designed in cooperation with the partners from University of Koblenz and University of Nottingham. It contains the orchestration of the application-level activities, calls to web services, and user interactions. Figure 2.1 gives a high-level overview of all parts of the intrinsic process on the application layer.

The technical subprocesses show the execution of security-specific enhancements of the business process to support the coupling with the TAS^3 security and trust framework.

In the following sections we will first detail the business process at the application level, then enhance the process with security annotations. During the further project time these annotations will lead to automatic realization of the annotations. A first realization is to enhance the business process with respective tasks, user interactions, and calls of security web services. This is presented in more detail in Section 2.4.

Enhancements of the business process are presented as BPMN diagrams. Components of the TAS^3 architecture are involved in the execution of the business process. They are part of further technical sequences presented as UML sequence diagrams. There are the following security-related sequences:

1. SSO Phase – Single sign-on.
2. Process Instantiation Sequence – Sequence flow for starting a business process (e.g. the Nottingham Process).
3. Human Task execution – Sequence flow for executing a human task.
4. Call to a Web Service – Sequence flow for calling a web service.
5. Web Service – Internal sequence flow of a web service on the application layer, service-specific (out of focus of this deliverable).

2.2 Modelling the Student Placement Process

The BPMN diagram of the student placements process contains the business-related sequence flows and data exchanges. When executing the process, more technical sequences shown in the following sections come into play.

The high-level overview of all parts of the process in Figure 2.1 contains four phases which are detailed on the application level in the following.

As next modelling step the business analyst will be supported to annotate the process with security annotations. The annotated process will later be used to generate a business process and configure the process and the security policy of the business process automatically as far as possible.

2.2.1 Student Placement Process Model

Figure 2.2 shows the process on the application level: The student starts the process and basic data is inferred from his/her login using single sign-on. Then he/she agrees to the TAS^3 terms and conditions, chooses a placement programme and registers for it.

Then service selection will take place. First, the discovery service is used to find available services. The user can choose from these services, for example based on their trust and reputation ranking. In a later version we plan to allow further automating this step with configuration parameters at the business process design level.

The next phase is service execution. The payload service is called, and the results are presented to the user.

In the last phase the user is presented a receipt, compiled from the audit records produced during process execution. He/She also has the opportunity to give feedback, which is transmitted to the trust feedback service, which uses it to update behavioural trust metrics.

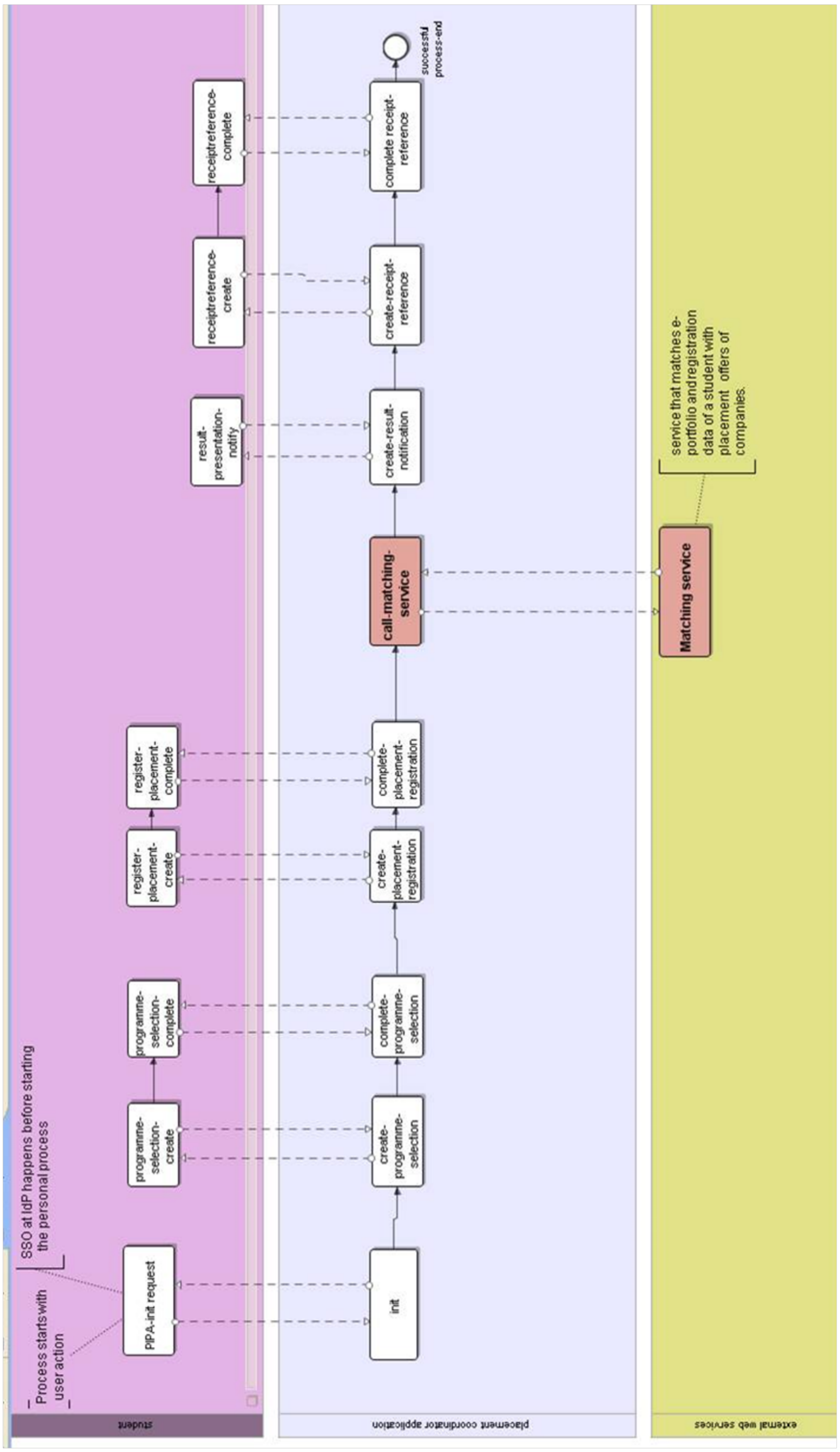

student
placement coordinator application
external web services
Process starts with user action
SSO at IdP happens before starting the personal process
PIPA-init request
init
programme-selection-create
create-programme-selection
programme-selection-complete
complete-programme-selection
register-placement-create
create-placement-registration
register-placement-complete
complete-placement-registration
call-matching-service
Matching service
service that matches e-portfolio and registration data of a student with placement offers of companies.
result-presentation-notify
create-result-notification
receiptreference-create
create-receipt-reference
receiptreference-complete
complete receipt-reference
successful process-end

Figure 2.2: Nottingham Student Placement Process Model

2.2.2 Security Annotated Process Model

Deliverable D3.1 ([8]) contains a first version how to annotate a BPMN process model with security annotations. The annotations can be related to all BPMN elements in the graphical BPMN modelling tool. An annotation starts with the security category, e.g. authorization, confidentiality, auditing, and then gives further parameter or details, e.g. the role or the object. Applying it to the concrete Student Placement Application resulted in two new categories, the trust category and the requestToUser category, which allows to describe that the process requests the user for further information. The annotations used in this demonstrator correspond to the version of end of second project year. We have learned from this demonstrator which resulted in a revised version of annotations, which we used for the next demonstrators described in the following chapters.

Figure 2.3 comprises the annotated student placement process.

The *audit* annotation of the executable pool ”placement coordinator application” means that the protocol of all relevant transactions, i.e. the message flow, will be sent to the audit bus. The user can then inspect the logging information on the portal.

The *AcceptPrivacyTerms* annotation results in having an interaction with the name role, i.e. ”student” for accepting the TAS^3 privacy information during the task with starts the whole process, because the annotated BPMN element is the task ”PIPA-init-request” which is the initializing task of the process. This name reflects that a user starts the process, see ”PIPA” in the glossary. The same task is also annotated with an authorization annotation. The meaning is, that the student (that is the role of the purple user pool) has to delegate the read right for his/her data to the process, that is the data from the Identity provider transferred during single sign-on.

The name of the user pool (in purple) is student. The annotation related to this pool is a *role* annotation which means that ”student” will be a security role during the process. Therefore, the assignment of a user to that role needs to be checked.

More annotations are related to the call of the matching service. Calling a web service is possible with several forms. Calling a web service in a security-related context should discover a service that fits to the predefined policies of the caller. After discovery the first service according to the trust ranking could be automatically chosen or the user will get the possibility to interact with the process and select a service. The user can use a trust policy for the service provider of the service called.

Therefore, a lot of annotations relate to the service call. We describe these annotations in the following.

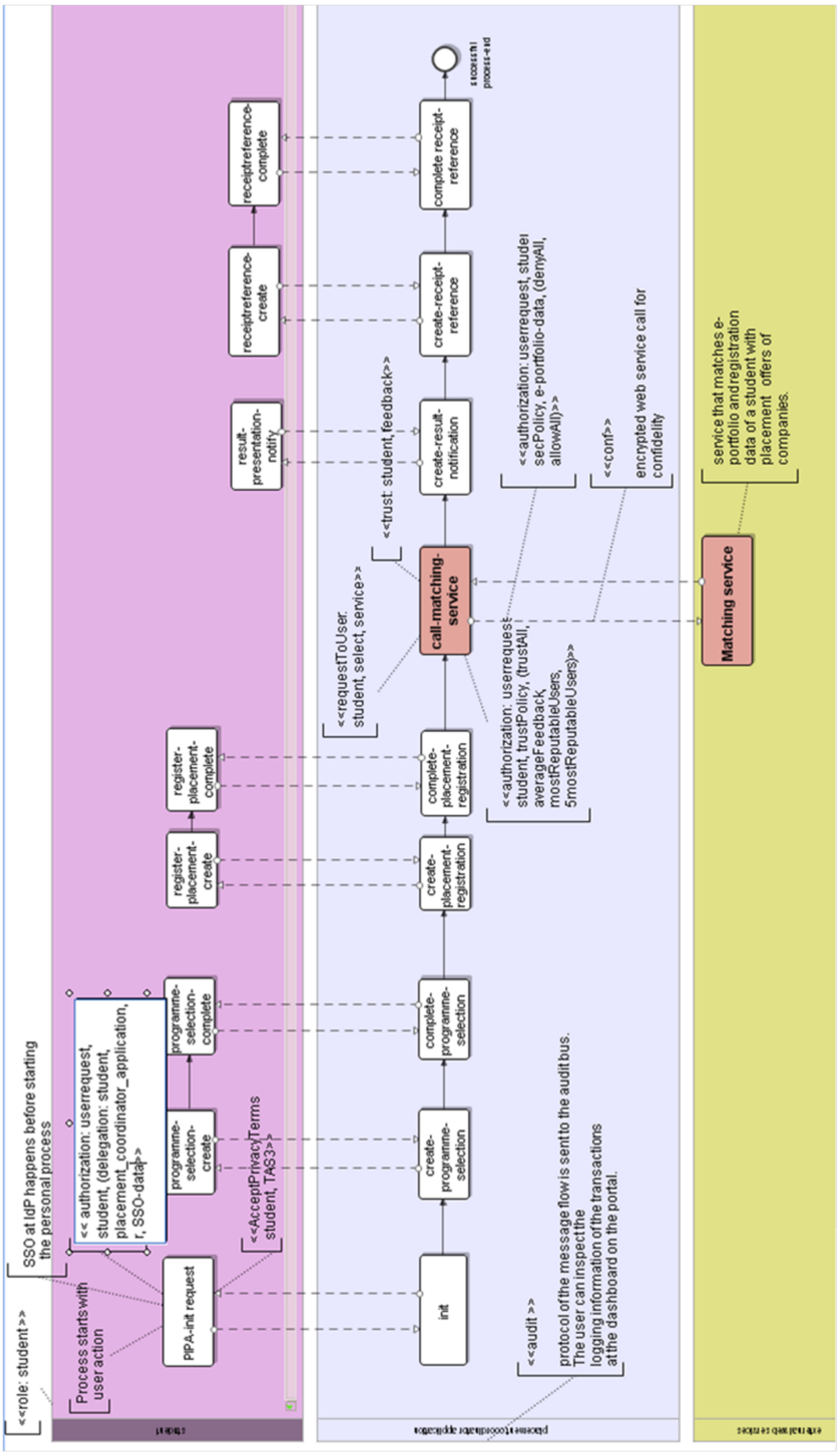
The *requestToUser* annotation defines that the selection of a service out of the result list of the discovery is done via interaction with the student.

The *authorization* annotations define that there is an interaction with ”student” (because ”userrequest” is given as parameter) for selecting a trust policy and a security policy for the e-portfolio data used by the web service to be called. There is also a set of trust policies and security policies specified in the annotation itself, which means that the user gets this list for selection. To offer a fixed set of policies is one form of supporting the user. Another possibility could be, that the user may define its own policy, which is more difficult for the user to do. This annotation will result in an interaction with the user selecting a security policy and a trust policy and then in a service discovery which uses the chosen trust policy for selecting services (of the service type) and a web service call also sending the chosen security policy.

The *conf* annotation defines that the web service call should be encrypted to guarantee confidentiality.

Finally, there is a *trust* annotation which means that after the web service call there should be an interaction with ”student” to let him/her give feedback on the web service execution. This trust feedback will have an effect to the trust ranking of the service provider of that web service.

The annotations should be handled automatically as far as possible, in future. For this demonstrator, we have managed the security and trust annotations in different ways, but manually. As an example the trust feedback annotation results in a subprocess with user interaction to give feedback and a web service call to the trust handling component for inserting the feedback. The next sections describe the handling of the trust and security properties in this demonstrator in detail.



<<role: student>>
Process starts with user action
SSO at IdP happens before starting the personal process
<< authorization: userrequest, student, (delegation: student, placement_coordinator_application, r, SSO-data)>>
<<AcceptPrivacyTerms student, TAS3>>
PIPA-init request
programme-selection-create
programme-selection-complete
register-placement-create
register-placement-complete
result-presentation-notify
receiptreference-create
receiptreference-complete
init
create-programme-selection
complete-programme-selection
create-placement-registration
complete-placement-registration
call-matching-service
create-result-notification
create-receipt-reference
complete receipt-reference
<<requestToUser: student, select, service>>
<<trust: student, feedback>>
<<authorization: userrequest, student, trustPolicy, (trustAll, averageFeedback, mostReputableUsers, 5mostReputableUsers)>>
<<authorization: userrequest, studen secPolicy, e-portfolio-data, (denyAll, allowAll)>>
<<conf>> encrypted web service call for confidelity
<<audit>> protocol of the message flow is sent to the audit bus. The user can inspect the logging information of the transactions at the dashboard on the portal.
Matching service
service that matches e-portfolio and registration data of a student with placement offers of companies.

Figure 2.3: Nottingham Student Placement Process with Security and Trust Annotations

2.3 Operational sequence and involved components

The following business process related components have been developed and are ready for integration:

- Business Process Execution Engine (T3-BP-ENGINE-ODE)
- GUI based on Intalio Tempo (T3-BP-MGR)
- Policy Information Point - Instance Roles (T3-BP-PIP-IR)
- Process Security Configuration Component (T3-BP-SM)
- Policy Enforcement Point - Requester (T3-PEP-RQ)

For integration purposes we also work with the Intalio Designer business process modelling tool (T3-EXT-INTALIO-DESIGNER). All components developed by Karlsruhe are described in detail in Deliverable D3.2.

We also had to integrate several components developed by other TAS^3 project partners:

- Audit Bus (T3-BUS-AUD), developed by NOT, message exchange format by UNIKOLD
- Matching Service (T3-SP-MATCHER), developed by NOT
- Trust Policy Decision Point (T3-PDP-T), developed by TUE
- ZXID (T3-ZXID-SRC), developed by SYM
- Trust Feedback Service (T3-TRU-FB), developed by KARL (WP5)

Deliverable D9.1, 2nd iteration, ([3]) contains an overall sequence diagram of the UK employability demonstrator (Figure 2.4). It shows the entire course of actions with focus on the security components while hiding the involved business process components.

The diagram can be divided into four phases: The Login Phase, the Service Selection Phase, the Web Service Call Phase and the End Phase (not explicitly depicted).

In the Login Phase the user wants to access the business process management GUI which is integrated into the portal. To this end, the user must be authenticated by a single sign-on (SSO) login. This functionality is provided by ZXID accessing an Identity Provider (IdP).

After logging in, the user selects an eligible programme and registers for it. For protection purposes of his/her data he/she specifies a trust and a security policy. Respecting the policy, the service discovery function of ZXID is used to discover appropriate Matching Services. From the list of discovered services the user can choose one for use.

The web service call of the Matching Service must be authorized by the PEP Requestor Out and the PEP Responder In. The result of the Matching Service call is presented to the user.

The user is able to trace the transactions of his/her process on the Dashboard. At the end of the process he/she has the chance to give feedback on the degree of satisfaction with the execution of the called service.

In the following we will take a closer look at the actions taking place during each phase. Differentiating between the process layer and the implementation layer we will clarify and detail the demonstrator sequences with respect to the business process support for security.

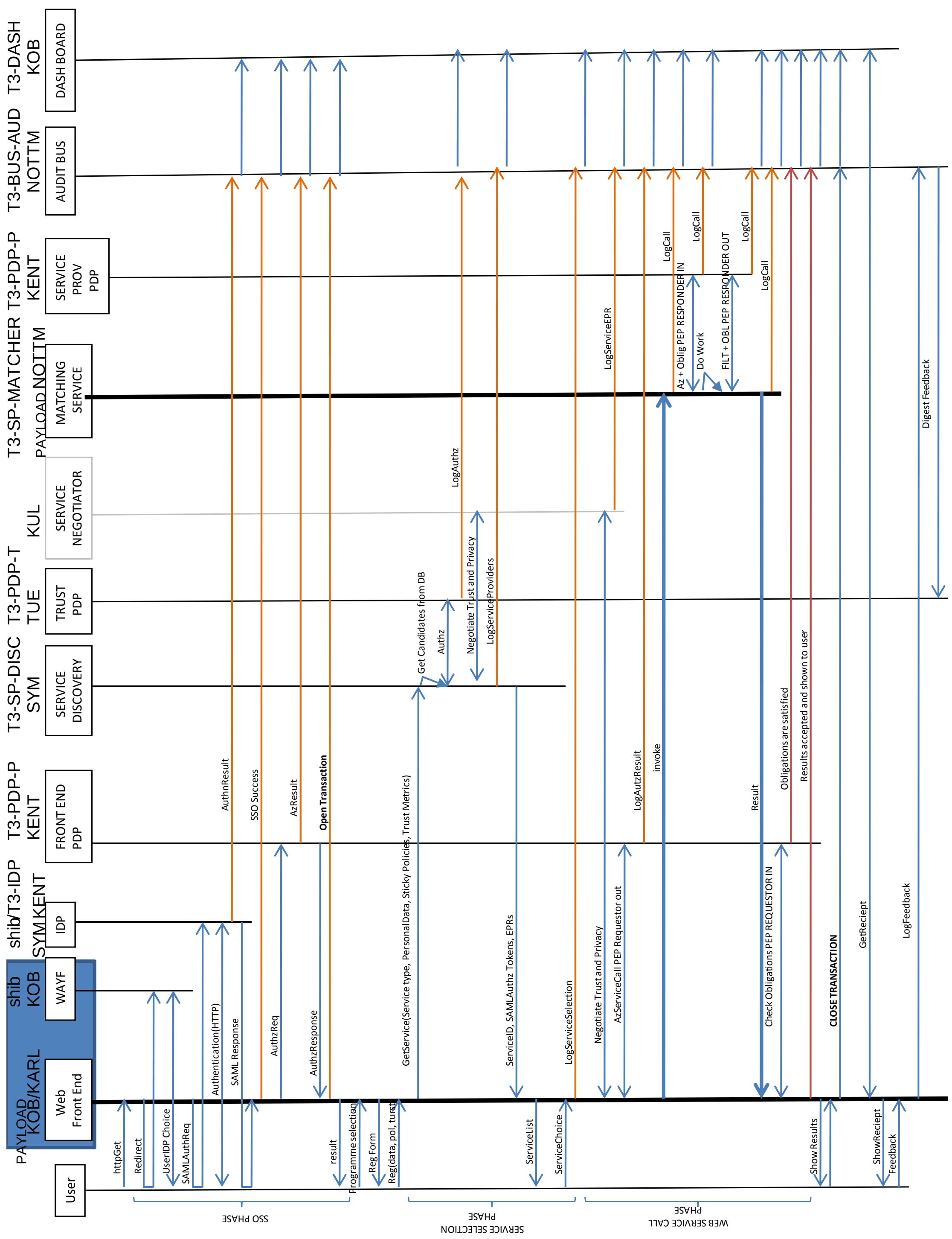


Figure 2.4: UK employability demonstrator sequence diagram

2.4 Implementation

In Section 2.2.2, we presented a security-enhanced process model. Based on the software components available, we described the technical protocol interactions between these components in Section 2.3.

A goal of WP3 is to use BPMN annotations like those presented in Section 2.2.2 to automatically create an executable process that fulfills the security properties defined by the annotations. However, such a comprehensive approach is not yet possible at the current stage of implementation and integration.

There are different ways to implement the security enhancements to the application process. We will assess for each phase of the process which alternative to pursue. Possible implementations are:

- The application business process, designed as a BPMN diagram and automatically compiled into executable BPEL. This includes GUI screens defined by XForms.
- Extensions to the BPEL engine, e.g. filters in the protocol stack. However, currently no such extension exists.
- Extensions to the business process GUI.
- External security components available as web services. This includes the T3-PEP-RQ and T3-BP-PIP-IR components described in Deliverable D3.2 ([9]).
- Additional tasks in the BPMN diagram of the business process that perform security-related tasks or call security components available as a web service.
- Stand-alone components coupled more loosely than by a synchronous web service call, e.g. by a publish/subscribe style event bus.

2.4.1 Single sign-on Phase

The single sign-on happens before user interacts with a business process instance. Figure 2.5 shows the protocol exchange when the user logs into in the business process user interface in order to start a new instance of a business process, or to continue working on an already running instance.

SSO capability is provided by a servlet running in the same servlet container as the business process user interface. This servlet handles the entire single sign-on protocol exchange. Any attributes pushed to it by the IdP are stored internally by ZXID and can be accessed via the ZXID session ID. The SSO servlet passes the ZXID session to the business process GUI by storing it in a HTTP cookie and redirecting the user. The business process GUI (Intalio Tempo, part of the T3-BP-MGR component) needs to be modified to understand the cookie set by the SSO servlet. No changes to the business process are necessary, as signon is handled completely independent from any running business process instance.

Figure 2.6 shows the sequence flow of a process instantiation.

Providing an XForm to users to let them start a business process instance is an existing mechanism in Intalio Tempo and the Apache Ode BPEL engine (T3-BP-ENGINE-ODE). Using this mechanism, a user id defined internally in the configuration of Intalio Tempo is transmitted to the BPEL engine. Currently, authorization for starting the process is handled using the existing security mechanism of Intalio Tempo. It is planned to replace this by authorization according to a workflow policy model, i.e. by a call to a dedicated Business Process Policy Decision Point.

To summarize, the single sign-on phase uses the following components:

- ZXID for Single sign-on (T3-ZXID-SRC).
- The identity-provider, a stand-alone component described in Deliverable D2.1 ([10]).
- A modified business process user interface accepting login via SSO.
- A web service storing the mapping between user names and ZXID sessions (as part of T3-PEP-RQ).

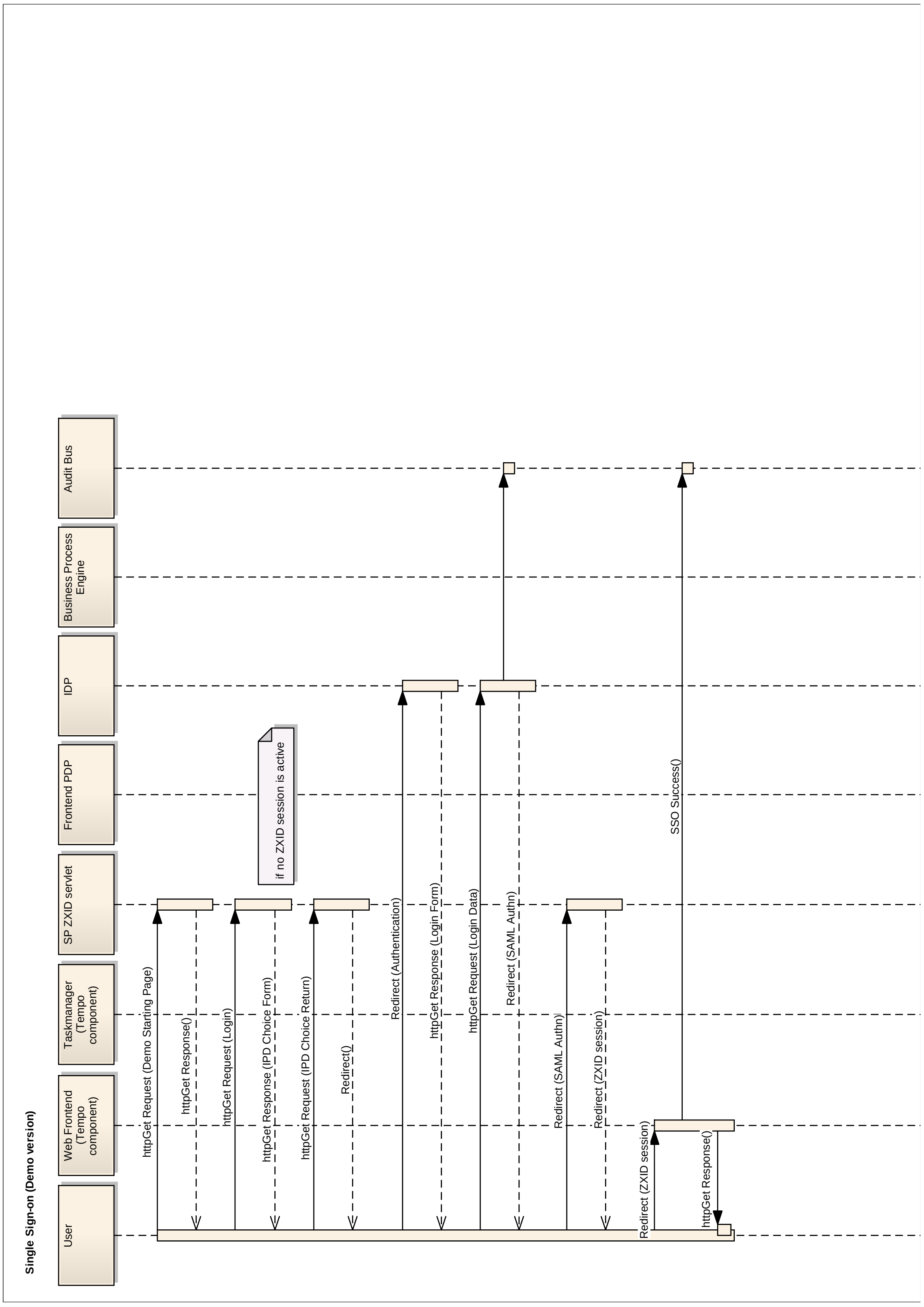

Single Sign-on (Demo version)
User
Web Frontend (Tempo component)
Taskmanager (Tempo component)
SP ZXID servlet
Frontend PDP
IDP
Business Process Engine
Audit Bus
if no ZXID session is active
httpGet Request (Demo Starting Page)
httpGet Response()
httpGet Request (Login)
httpGet Response (IPD Choice Form)
httpGet Request (IPD Choice Return)
Redirect()
Redirect (Authentication)
httpGet Response (Login Form)
httpGet Request (Login Data)
Redirect (SAML Authn)
Redirect (SAML Authn)
Redirect (ZXID session)
Redirect (ZXID session)
httpGet Response()
SSO Success()

Figure 2.5: Single Sign-on Sequence

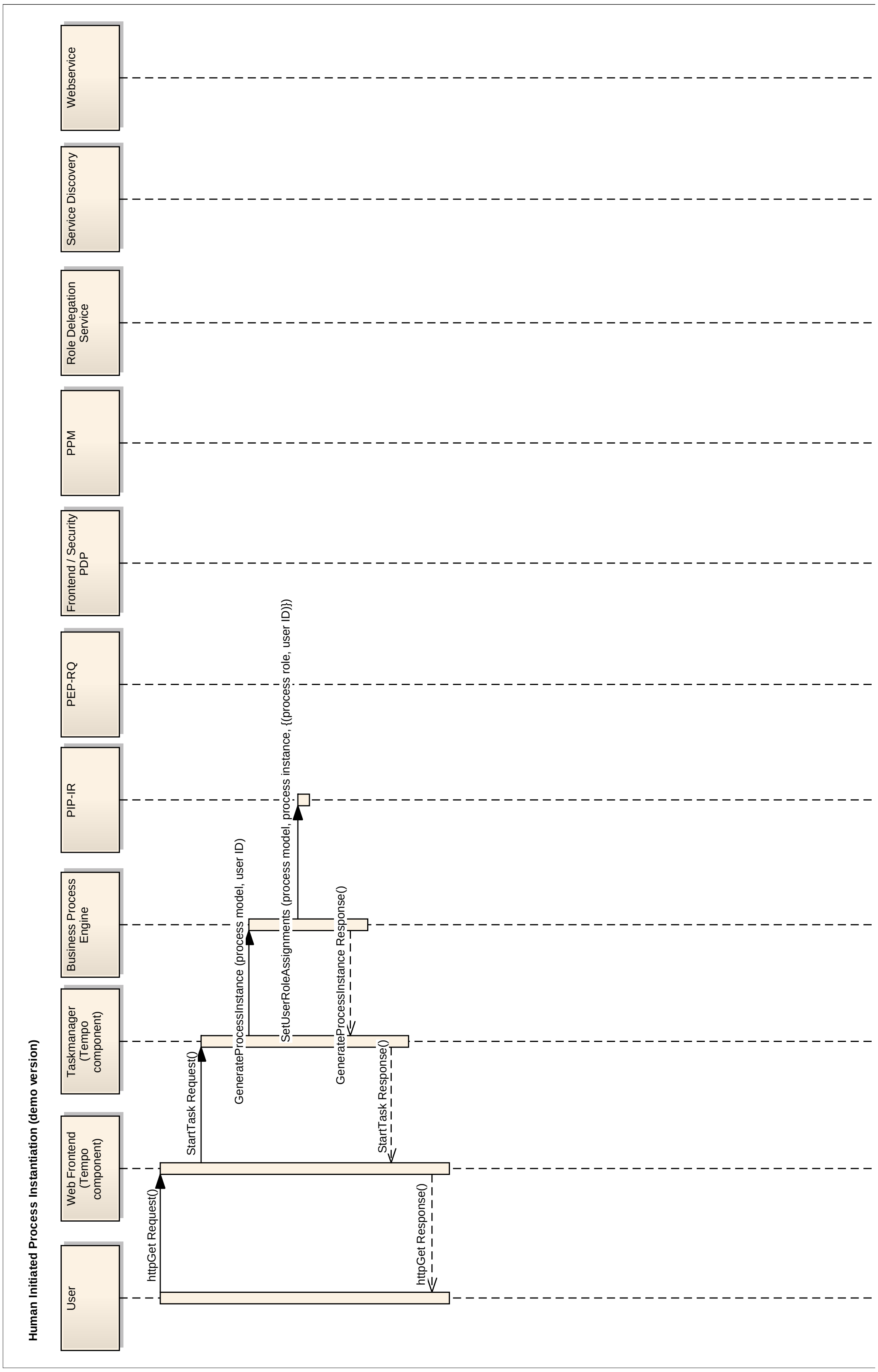

Human Initiated Process Instantiation (demo version)
User
Web Frontend (Tempo component)
Taskmanager (Tempo component)
Business Process Engine
PIP-IR
PEP-RQ
Frontend / Security PDP
PPM
Role Delegation Service
Service Discovery
Webservice
httpGet Request()
StartTask Request()
GenerateProcessInstance (process model, user ID)
SetUserRoleAssignments (process model, process instance, {(process role, user ID)})
GenerateProcessInstance Response()
StartTask Response()
httpGet Response()

Figure 2.6: Process Instantiation Sequence

2.4.2 Programme Selection and Registration

The Programme Selection and Registration phase can be divided into four subphases, depicted on Figures 2.7 to 2.10. The Process Instantiation and Disclaimer subphase starts a new process instance. The user has to accept the disclaimer with TAS^3 terms and conditions for handling privacy data to continue the process.

In order to support the TAS^3 identity management system and enable the business process to access the attributes pushed by the IdP, we introduce two new functions in the T3-PEP-RQ component: First, the mapping from a username (used internally by Tempo and the process) to the ZXID session of that user can be set. Second, it is possible to query SSO attributes for a particular user.

Right after the user has agreed to the terms and conditions of the TAS^3 framework, he/she is registered as the holder of the student role in that process instance. This is a security mechanism that guarantees that no other user is allowed to perform tasks of this process instance. The registration interface is provided by the Policy Information Point - Instance Roles (T3-BP-PIP-IR), a web service component that stores the current assignment of users to process roles. In the current version of the process we also have web service calls for registering the process model, the role assigned to the human tasks and the process instance of the current process instance (see Figure 2.7). In later versions of the demonstrator we will remove these calls from the application layer, i.e. from the process model, and integrate them into the ODE process engine on the security enforcement layer.

In the SSO Attribute subphase the personal data of the user submitted by the ZXID SSO login is transferred to the process instance via web service calls to the PEP-RQ. The user has to agree on the usage of his/her SSO data by the business process (see Figures 2.8 and 2.9).

The selection of the placement programme the user wants to attend and the registration for this programme are implemented as pure business process tasks without using external components. This may change at a more elaborated stage of the application process (see Figure 2.9).

Within the last part of the registration process the user has to define a policy of his/her trust requirements. The trust policy is used to discover an appropriate service. Only services fulfilling the user's trust requirements are returned from discovery. This functionality is provided by the T3-PEP-RQ component, which uses the ZXID library and its `zxid_get_epr()` function to discover all services of the relevant service type that are available. These services are then checked by the Trust PDP (T3-PDP-T) to determine whether the service fulfills the user's trust policy, and if so, the value of the service's trust ranking is returned (see Figure 2.10).

Nottingham provides an Audit Bus for logging, which allows to present the logging information on the Portal in a so-called Audit Trail Viewer. We log all kinds of significant and security relevant events to the Audit Bus. As mostly the PEP-RQ is involved in the significant process tasks we integrated the logging into the PEP-RQ. This allows the PEP-RQ independently writing to the Bus as well as the process calling the PEP-RQ as a proxy to send logs to the Audit Bus.

To summarize, the Programme Selection and Registration phase uses the following components:

- A web service allowing the retrieval of a user's attributes from the ZXID session (also as part of T3-PEP-RQ).
- A web service storing the assignment of users to roles in a business process instance (the existing T3-PIP-IR).
- An additional security-related task in the business process calling the T3-PIP-IR.
- Additional security-related tasks in the business process collecting the user's policy choice and calling the discovery function in the PEP-RQ.
- A web service (as part of T3-PEP-RQ) for performing service discovery and compiling the results.
- A stand-alone Trust PDP (T3-PDP-T).

Human tasks take place several time during the process flow. On the implementation layer a couple of interactions happen not being visible on the process model layer. Figure 2.11 shows the sequence flow of the execution of a human task. Most of the interactions happen between the Web Interface as part of the Intalio Tempo component (part of the T3-BP-MGR), the Taskmanager (also part of Intalio Tempo) and the business process engine (T3-BP-ODE). We integrated the call to the PIP-IR to check if the user is allowed to perform the current human task.

Figure 2.12 shows the sequence flow of discovering services according to the user's trust policy. The call to the security PDP (Policy Decision Point) is not implemented yet. We plan to integrate it as soon as relevant security policies are used in the application. The logging activities to the Audit Bus are not explicitly depicted in all the sequence diagrams in this deliverable.

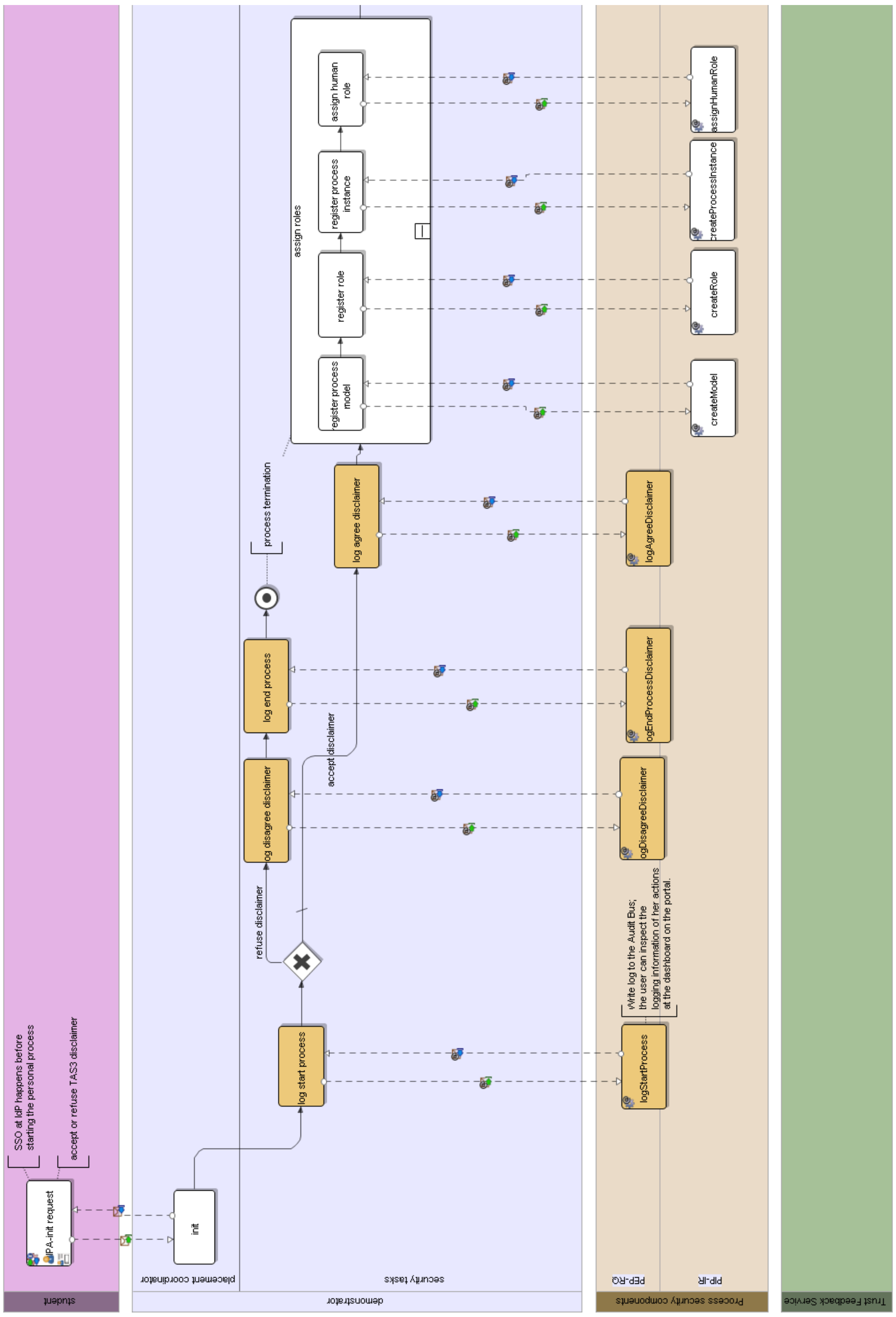
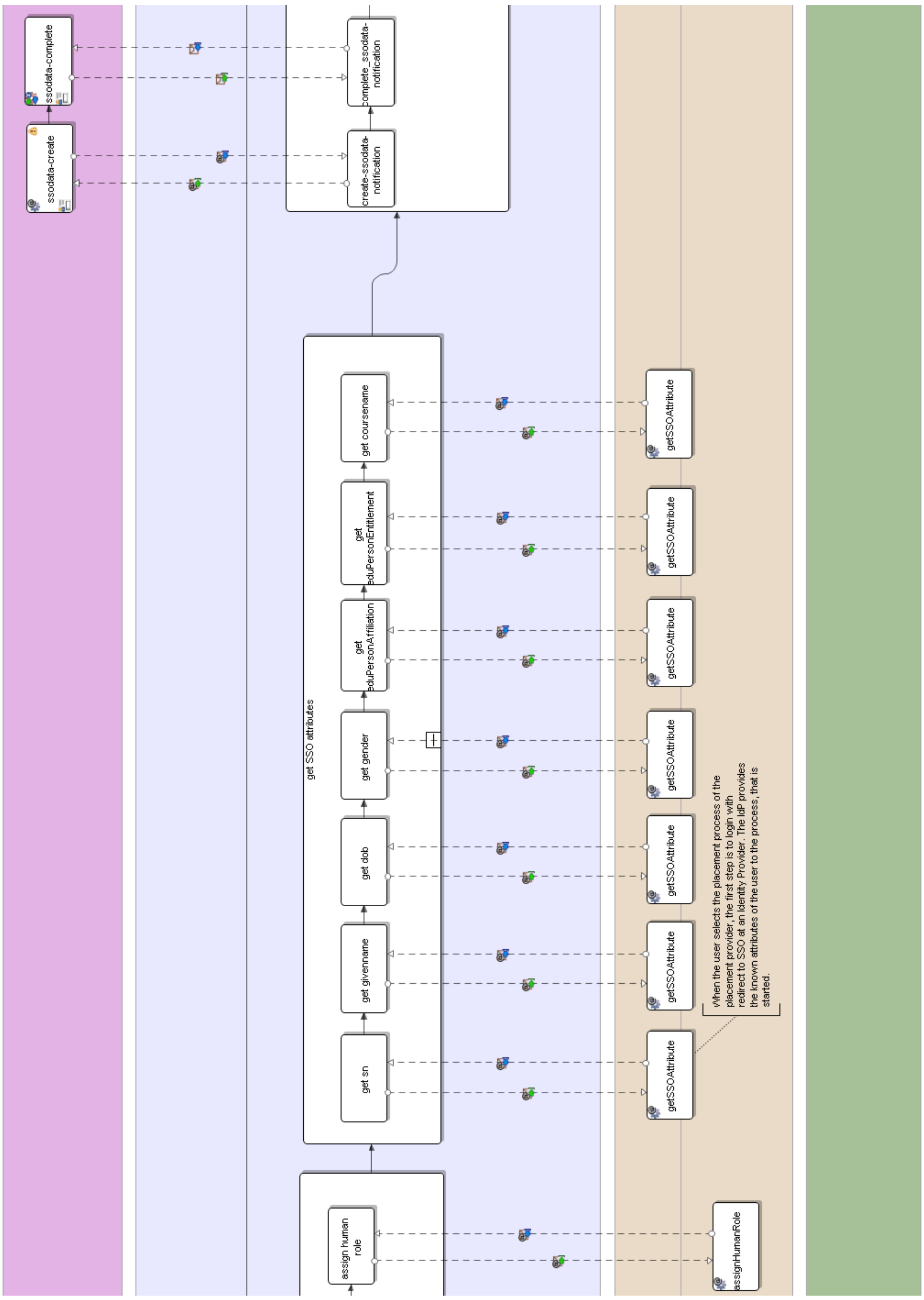
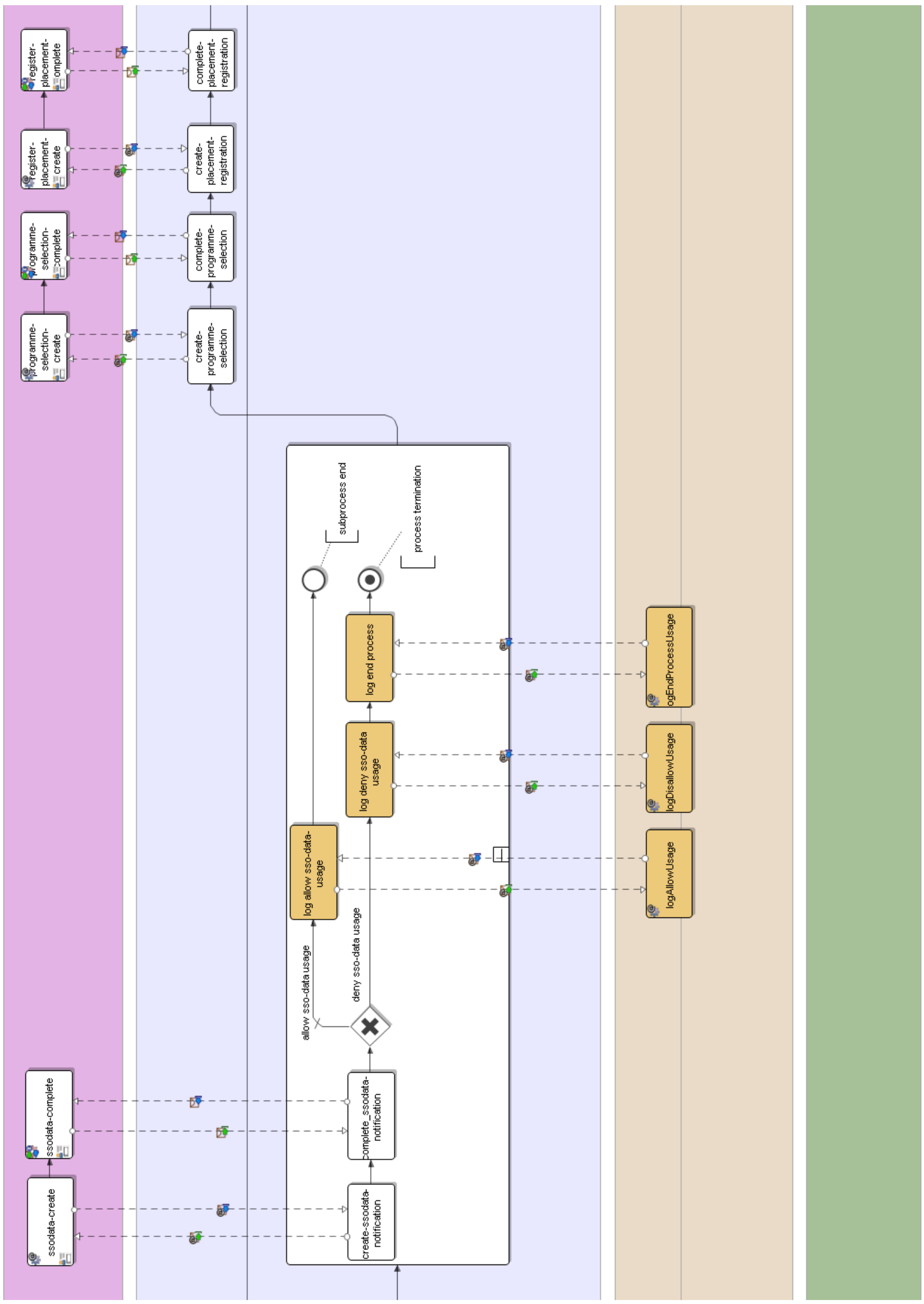


Figure 2.7: Programme Selection and Registration - Part 1



When the user selects the placement process of the placement provider, the first step is to login with redirect to SSO at an Identity Provider. The IdP provides the known attributes of the user to the process, that is started.

Figure 2.8: Programme Selection and Registration - Part 2



register-placement-complete
register-placement-create
programme-selection-complete
programme-selection-create
complete-placement-registration
create-placement-registration
complete-programme-selection
create-programme-selection
subprocess end
process termination
log end process
log deny sso-data usage
log allow sso-data-usage
deny sso-data usage
allow sso-data usage
logEndProcessUsage
logDisallowUsage
logAllowUsage
ssodata-complete
ssodata-create
complete_ssodata-notification
create-ssodata-notification

Figure 2.9: Programme Selection and Registration - Part 3

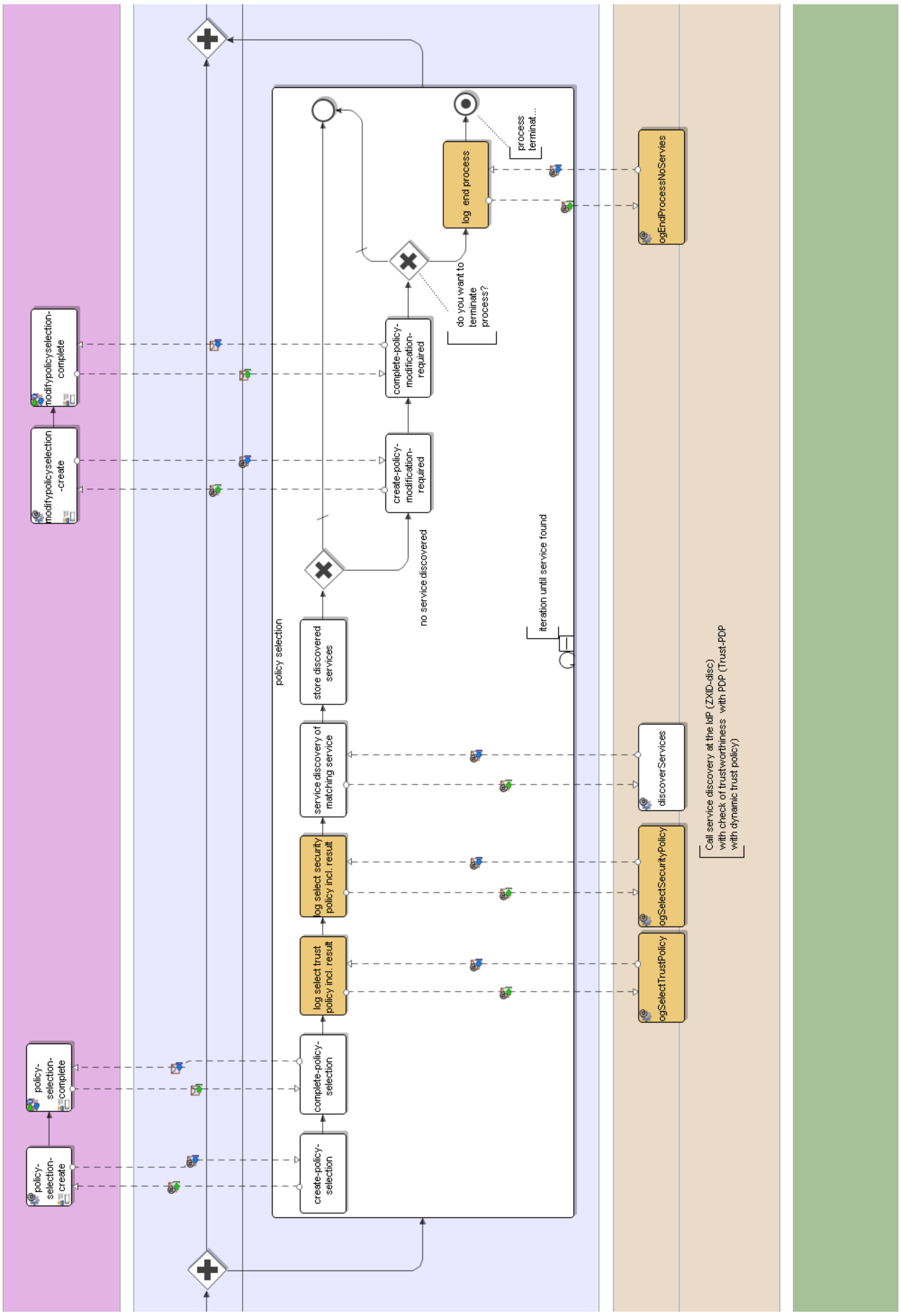

policy-selection-create
policy-selection-complete
modifypolicyselection-create
modifypolicyselection-complete
policy selection
create-policy-selection
complete-policy-selection
log select trust policy incl. result
log select security policy incl. result
service discovery of matching service
store discovered services
no service discovered
create-policy-modification-required
complete-policy-modification-required
do you want to terminate process?
log end process
process terminat...
iteration until service found
logSelectTrustPolicy
logSelectSecurityPolicy
discoverServices
logEndProcessNoServies
Call service discovery at the IdP (ZXID-disc) with check of trustworthiness with PDP (Trust-PDP with dynamic trust policy)

Figure 2.10: Programme Selection and Registration - Part 4

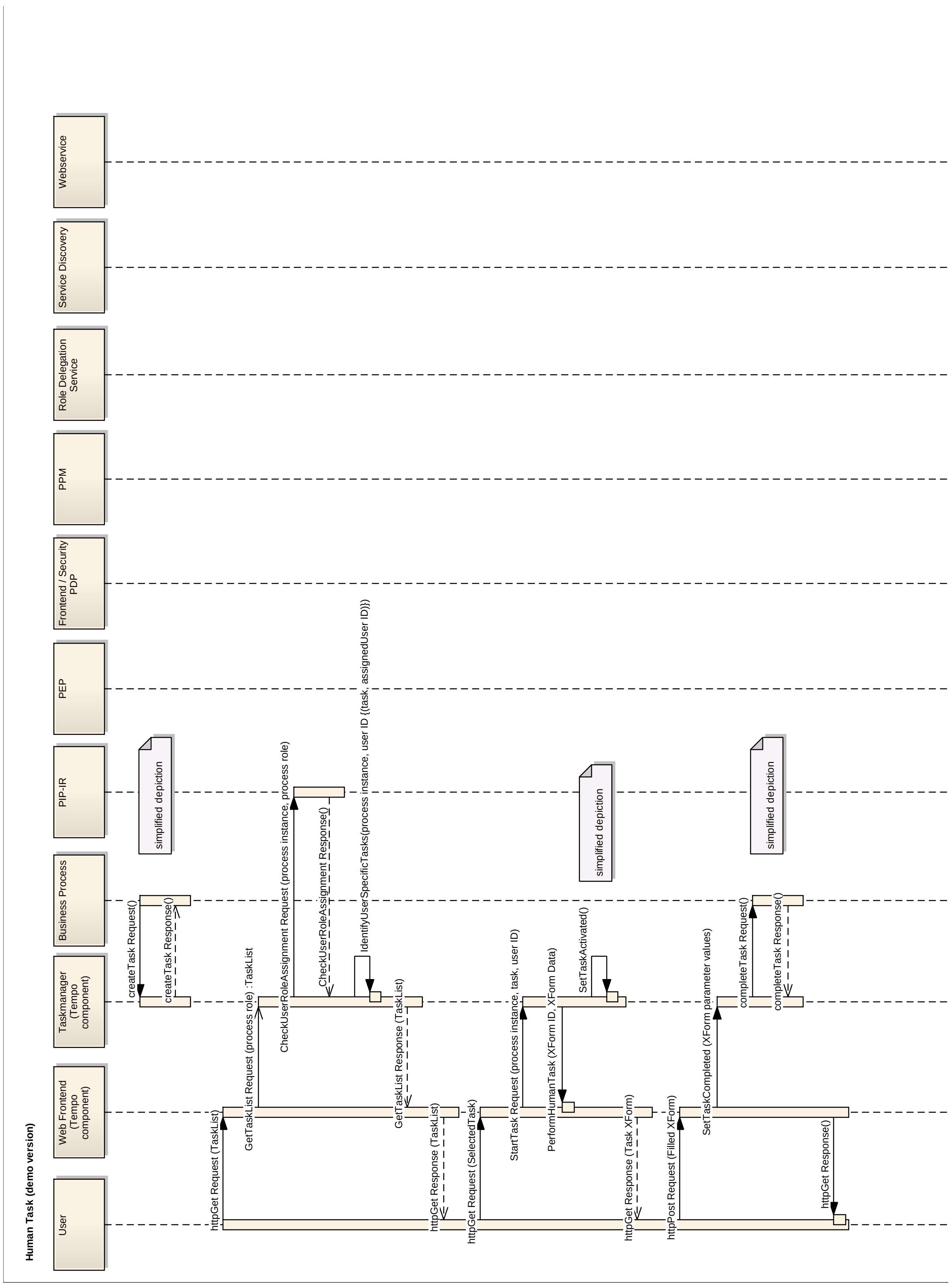

Human Task (demo version)
User
Web Frontend (Tempo component)
Taskmanager (Tempo component)
Business Process
PIP-IR
PEP
Frontend / Security PDP
PPM
Role Delegation Service
Service Discovery
Webservice
createTask Request()
createTask Response()
simplified depiction
httpGet Request (TaskList)
GetTaskList Request (process role) :TaskList
CheckUserRoleAssignment Request (process instance, process role)
CheckUserRoleAssignment Response()
IdentifyUserSpecificTasks(process instance, user ID {(task, assignedUser ID)})
GetTaskList Response (TaskList)
httpGet Response (TaskList)
httpGet Request (SelectedTask)
StartTask Request (process instance, task, user ID)
PerformHumanTask (XForm ID, XForm Data)
SetTaskActivated()
simplified depiction
httpGet Response (Task XForm)
httpPost Request (Filled XForm)
SetTaskCompleted (XForm parameter values)
completeTask Request()
completeTask Response()
simplified depiction
httpGet Response()

Figure 2.11: Human Task Sequence

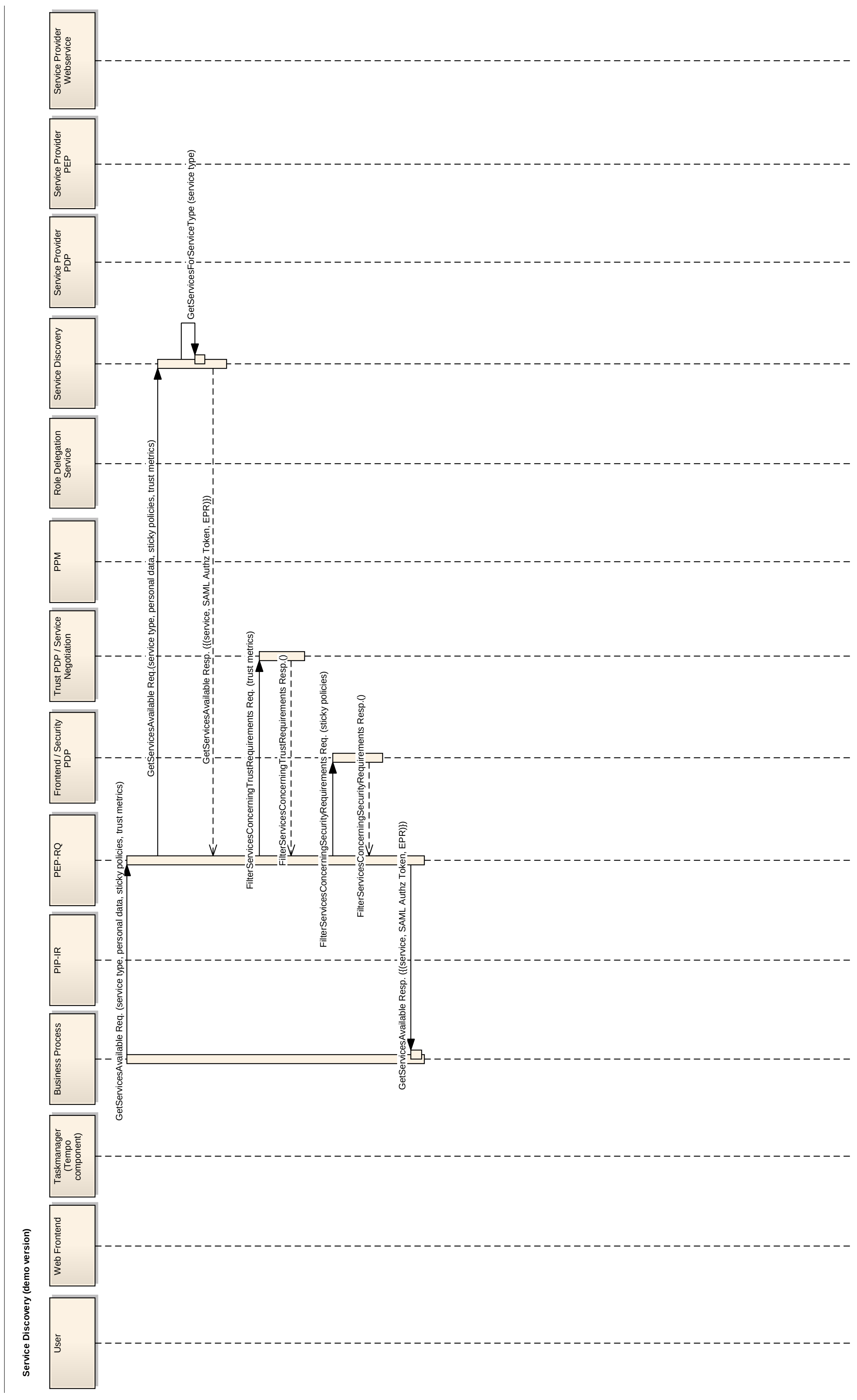

Service Discovery (demo version)
User
Web Frontend
Taskmanager (Tempo component)
Business Process
PIP-IR
PEP-RQ
Frontend / Security PDP
Trust PDP / Service Negotiation
PPM
Role Delegation Service
Service Discovery
Service Provider PDP
Service Provider PEP
Service Provider Webservice
GetServicesAvailable Req. (service type, personal data, sticky policies, trust metrics)
GetServicesAvailable Req.(service type, personal data, sticky policies, trust metrics)
GetServicesForServiceType (service type)
GetServicesAvailable Resp. ({(service, SAML Authz Token, EPR)})
FilterServicesConcerningTrustRequirements Req. (trust metrics)
FilterServicesConcerningTrustRequirements Resp.()
FilterServicesConcerningSecurityRequirements Req. (sticky policies)
FilterServicesConcerningSecurityRequirements Resp.()
GetServicesAvailable Resp. ({(service, SAML Authz Token, EPR)})

Figure 2.12: Web Service Discovery Sequence

2.4.3 Service Selection Phase

Special tasks in the process call the T3-PEP-RQ to retrieve the discovery results and display them to the user as web site in an XForm. When the user has chosen a service, it is registered in the T3-PIP-IR. This call passes through the PEP-RQ.

To summarize, the Service Selection phase uses the following components:

- Additional security-related tasks in the business process presenting the available services to the user and registering his/her choice in the T3-PIP-IR.
- A web service handling the assignment of endpoints to service types in a process (the existing T3-PIP-IR component).
- A web service acting as a proxy for registering the assignment (as part of T3-PEP-RQ).

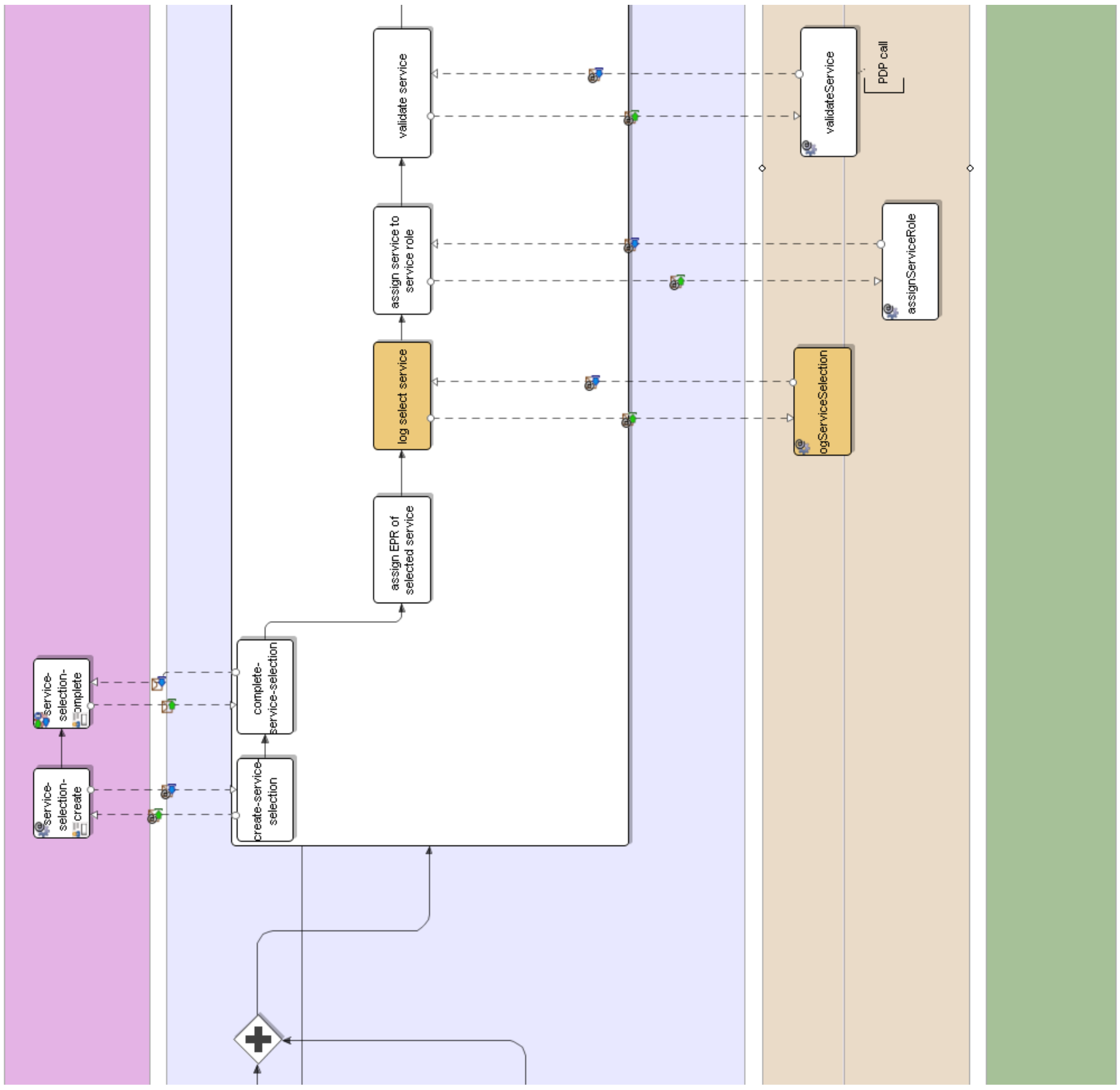


Figure 2.13: Service Selection

2.4.4 Service Execution

Because of possible delay between service discovery and service execution the trust level of the service to call has to be validated again before calling the service. This happens via calling the Trust PDP to determine whether the service still fulfills the user's trust policy. The PEP-RQ acts as a proxy for that web service call. If the trust requirements are still valid the service, in this case a matching service, is called. The PEP-RQ acts as a proxy for this call, as well. To call the payload service the PEP-RQ uses the ZXID function `zxid_call()`. This function encrypts and signs the SOAP message before performing the call.

To summarize, the Service Execution phase uses the following components and services:

- A web service acting as a proxy for validating the matching service call (as part of the PEP-RQ).
- A stand-alone Trust PDP (T3-PDP-T).
- A web service acting as a proxy for calling the matching service (as part of the PEP-RQ).
- ZXID for performing secure web service calls (T3-ZXID-SRC).
- The Matching Service (T3-SP-MATCHER).

Figure 2.16 shows the sequence flow of a web service call. The sequence flow takes place every time a service task, i.e. a web service or subprocess, is called. The diagram shows calls to the Security PDP which are not implemented yet. We plan to integrate them as soon as relevant security policies are available in the application. Regarding the call to the Trust PDP the diagram shows the case that the service call is valid, i.e. without a loop.

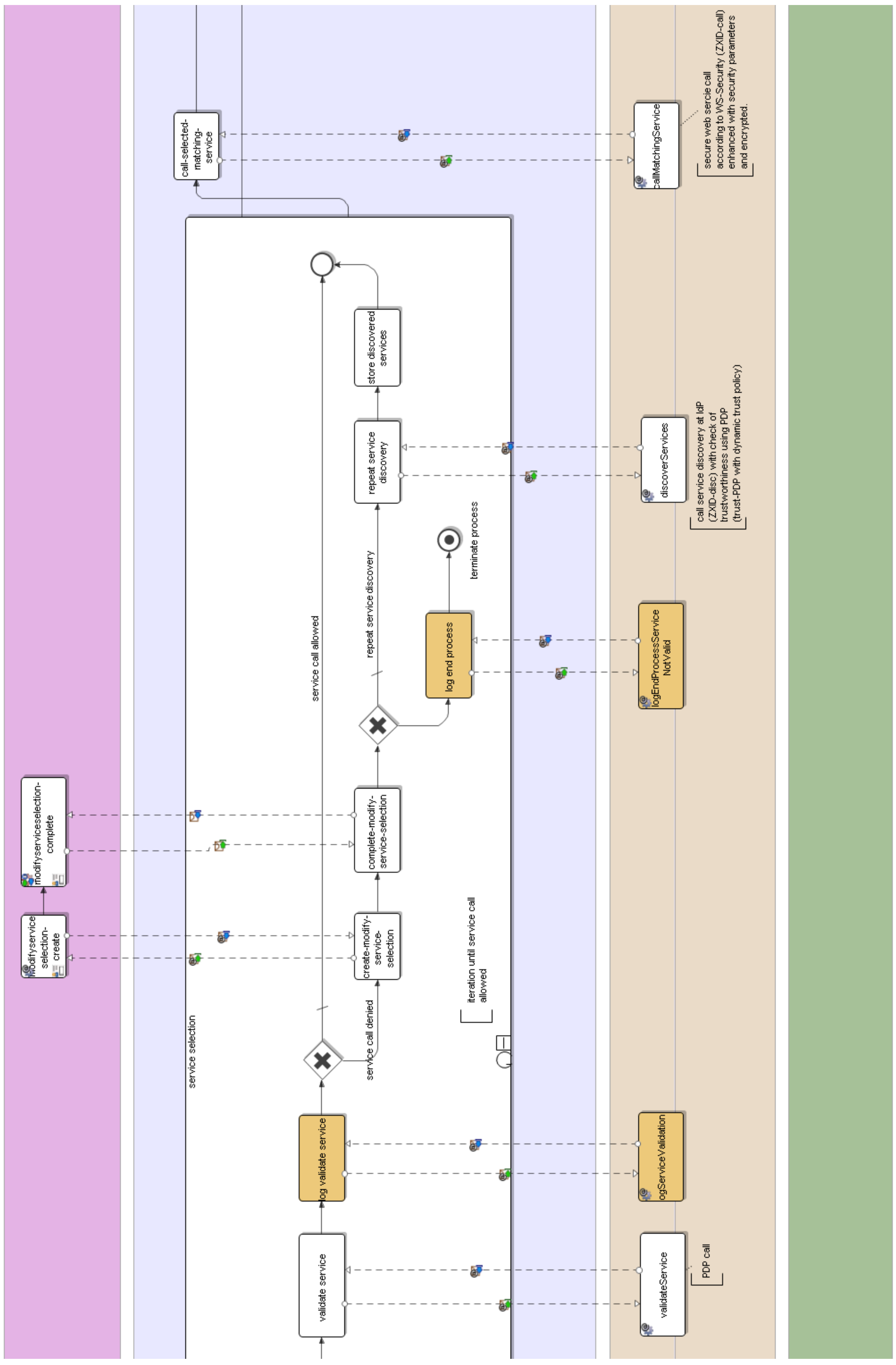

call-selected-matching-service
callMatchingService
secure web sercie call according to WS-Security (ZXID-call) enhanced with security parameters and encrypted.
store discovered services
repeat service discovery
discoverServices
call service discovery at IdP (ZXID-disc) with check of trustworthiness using PDP (trust-PDP with dynamic trust policy)
service call allowed
repeat service discovery
terminate process
log end process
logEndProcessService NotValid
modifyserviceselection-complete
complete-modify-service-selection
modifyservice selection-create
create-modify-service-selection
iteration until service call allowed
service selection
service call denied
log validate service
logServiceValidation
validate service
validateService
PDP call

Figure 2.14: Service Execution - Part 1

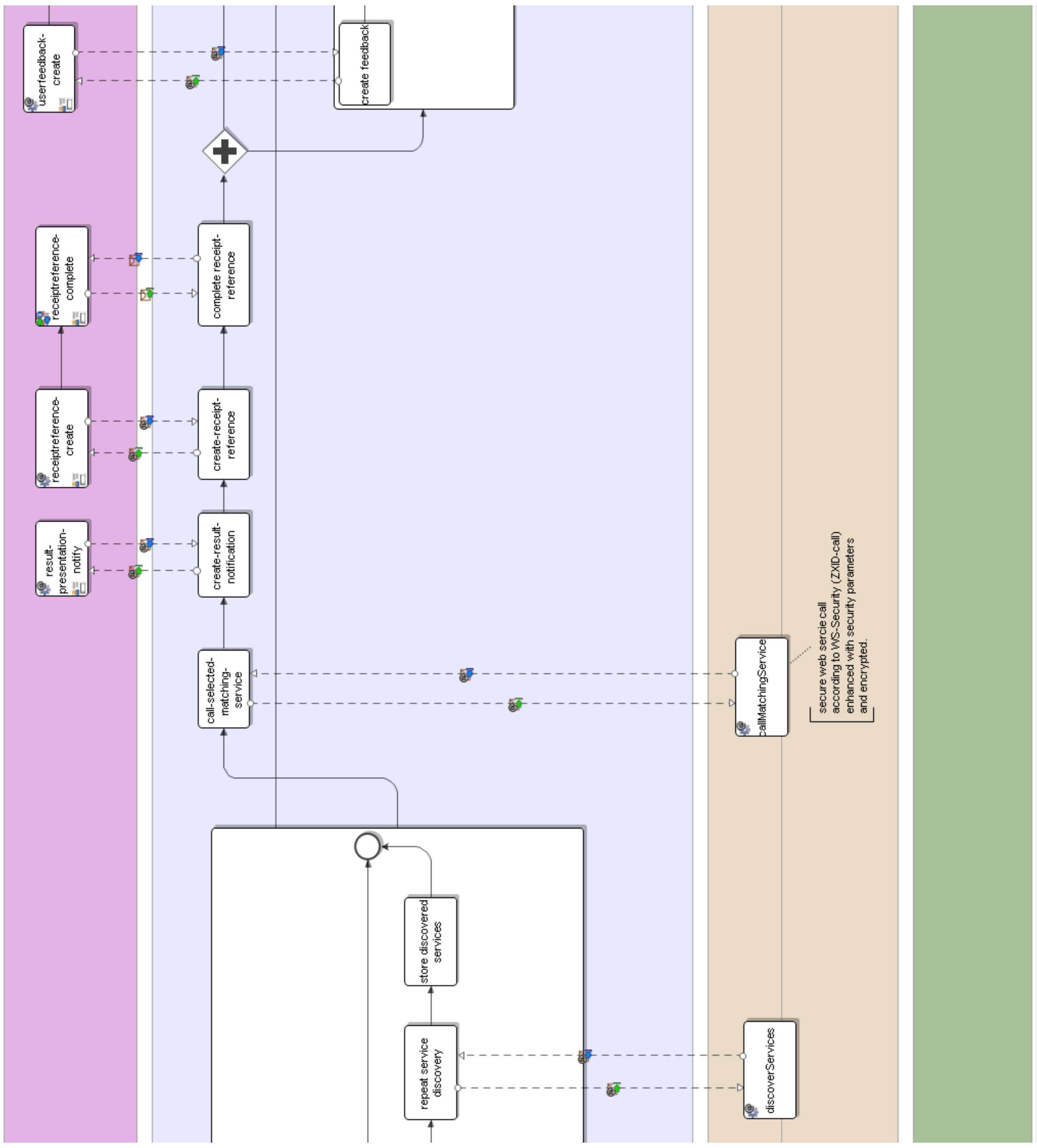

userfeedback-create
create feedback
receiptreference-complete
complete receipt-reference
receiptreference-create
create-receipt-reference
result-presentation-notify
create-result-notification
call-selected-matching-service
callMatchingService
secure web sercie call according to WS-Security (ZXID-call) enhanced with security parameters and encrypted.
store discovered services
repeat service discovery
discoverServices

Figure 2.15: Service Execution - Part 2

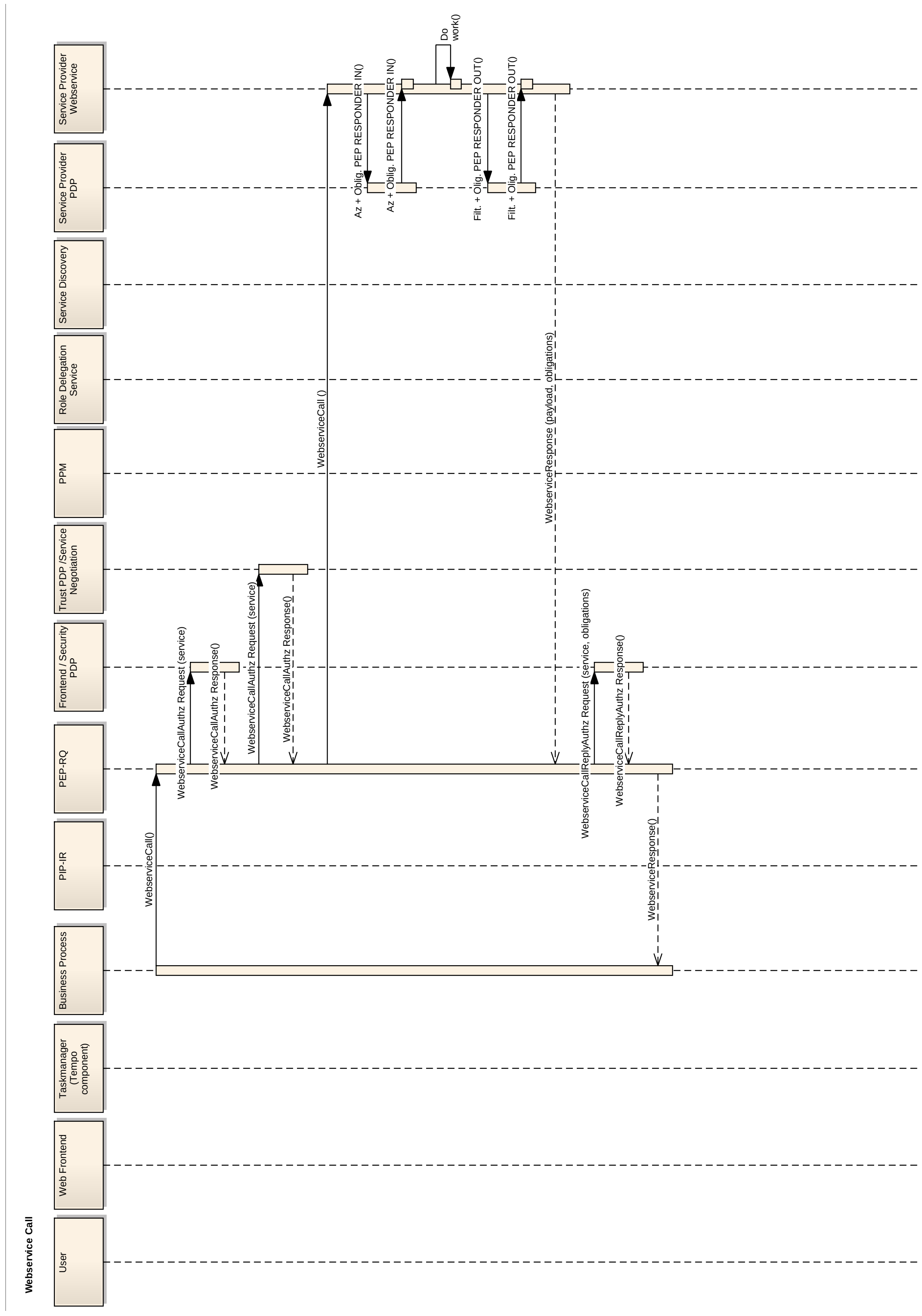

Webservice Call
User
Web Frontend
Taskmanager (Tempo component)
Business Process
PIP-IR
PEP-RQ
Frontend / Security PDP
Trust PDP /Service Negotiation
PPM
Role Delegation Service
Service Discovery
Service Provider PDP
Service Provider Webservice
WebserviceCall()
WebserviceCallAuthz Request (service)
WebserviceCallAuthz Response()
WebserviceCallAuthz Request (service)
WebserviceCallAuthz Response()
WebserviceCall ()
Az + Oblig. PEP RESPONDER IN()
Az + Oblig. PEP RESPONDER IN()
Do work()
Filt. + Olig. PEP RESPONDER OUT()
Filt. + Olig. PEP RESPONDER OUT()
WebserviceResponse (payload, obligations)
WebserviceCallReplyAuthz Request (service, obligations)
WebserviceCallReplyAuthz Response()
WebserviceResponse()

Figure 2.16: Web Service Call Sequence

2.4.5 End Phase

In the End Phase of the process the user has the chance to give feedback. There exists a web form for user interaction. This feedback is sent via a direct web service call to the Trust Feedback Service (T3-TRU-FB).

The End Phase uses the following components:

- The Trust Feedback Service (T3-TRU-FB).

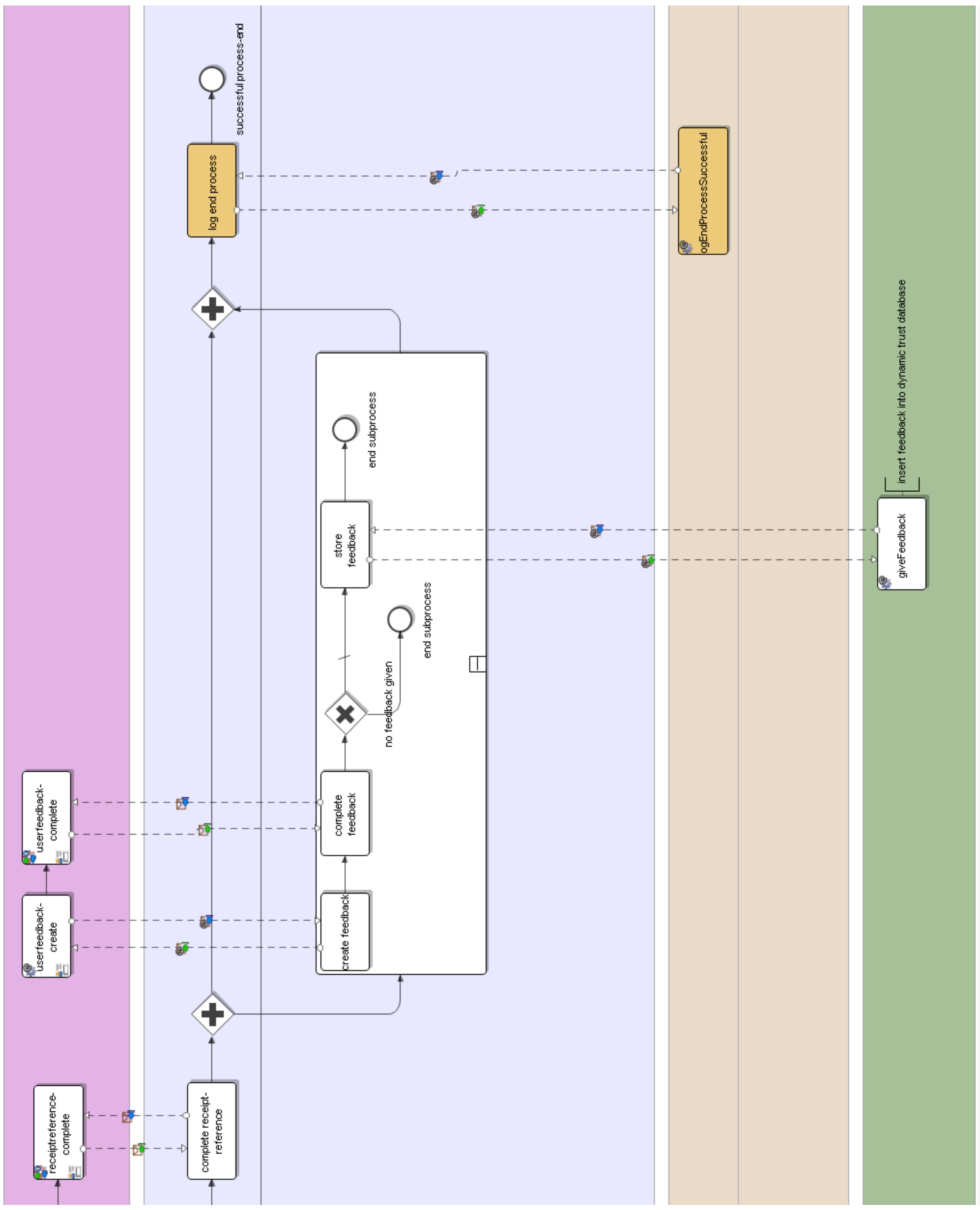


Figure 2.17: Process End Phase

3 The Nottingham Use Case of Student Placements – Second Demonstrator

3.1 Overview and Innovations of the second demonstrator

This section describes the second demonstrator for the Nottingham Student Placement use case.

Section 3.1 gives an overview of the improvements compared to the first Nottingham demonstrator of Chapter 2. Section 3.2 describes the approach for modelling the business process for the demonstrator. Section 3.3 introduces the components of the TAS^3 framework that are directly integrated into the business process. Section 3.4 details the business process implementation of the second demonstrator. Concerning WP3 the following improvements have been made compared to the first demonstrator:

- Refined business process security annotations
- Annotation transformation in business process policy elements and process model adaptations
- Refined assignment of users to tasks
- A business process policy in the demonstrator used for specifying constraints on the following security aspects:
  - Assignment of users to tasks
  - Delegation
- Refined trust and security policy handling
  - Delegation
- Integration of access to a personal data store (PDS) into the business process
- Improved user centricity
  - Refined security modelling
  - Enhanced business process GUIs
  - Automated process start via web service call.
- Improved audit logging

For the first demonstrator we already used business process security annotations to specifiy security requirements as an additional security layer to the graphical business process model. The syntax and semantics of these annotations have been considerably improved as well as the documentation, see the third iteration of Deliverable 3.1 ([1]).

Some parts of these annotations are being transformed into a business process security policy. As described in [1] we have improved functionality for the assignment of users to tasks. Assignment constraints defined as annotations (like binding of duty) become part of the business process policy and will be used by the business process PDP to decide upon requests regarding the assignment of users to tasks. For some annotations like deciding upon the acceptance of TAS^3 terms&conditions, we provide as transformation an adaption of the business process model. To this end, we use reusable process patterns with user interactions which will be automatically inserted into the process model before deployment. The patterns are all security-related process patterns because adequate subprocesses contain security-related functionality like consent to terms&conditions, giving trust feedback about service use, or selecting applicable data security policies. These tasks depend on the concrete process instance, e.g. the concrete user involved in the process, so we want to select a convenient process pattern according to this instance-specific status information of the process. That is why we provide process adaptation functionality to adapt the process to these security-related and instance-specific needs.

In the second Nottingham demonstrator two participants are involved: A learner and a placement coordinator. During the business process the coordinator initiates a web service call to the matching service. The matching service internally requires access to the learner's PDS to perform the matching. Therefore the call to the matching service must be executed in the name of the learner, but the call is triggered by the coordinator. This requires delegation from the learner to the coordinator. From the business process perspective delegation is supported by providing an appropriate annotation type on the modelling layer, providing a transformation of this annotation type into a part of the business process policy and by handling business process-specific delegation aspects by the Business Process Delegation Service (T3-BP-DEL).

Nottingham provides an adapted version of Mahara for use in the second demonstrator as personal data store. For integration into the TAS^3 architecture several new interfaces have to be developed. The customized Mahara system supports defining data policies for data of the PDS and enforcing it within the personal data store.

In the second demonstrator a main focus of attention concerns user centricity aspects. The modelling of security constraints has been specified in more detail and appropriate documentation has been created. Developing an annotation assistance tool based on ontologies is ongoing work in cooperation with our project partner VUB, and is planned to be used in the next period demonstrator.

In cooperation with Nottingham and Karlsruhe Koblenz adjusted the user interfaces to be applicable for the business process user with privacy concerns, i.e. the learner. Typically, he/she is not expert in using business processes, as e.g. the holder of the coordinator role, and expects to be guided through the process by the process application itself.

To hide technology aspects from the end-user, the second demonstrator business process is automatically started by a web service call initiated from the placement administrator website. This redirects the learner directly to the business process user interface, which is integrated into the Dashboard.

The logging of the audit events will also be improved. The events to be logged will be stronger coordinated between the project partners, and therefore the reporting for the user will be more informative.

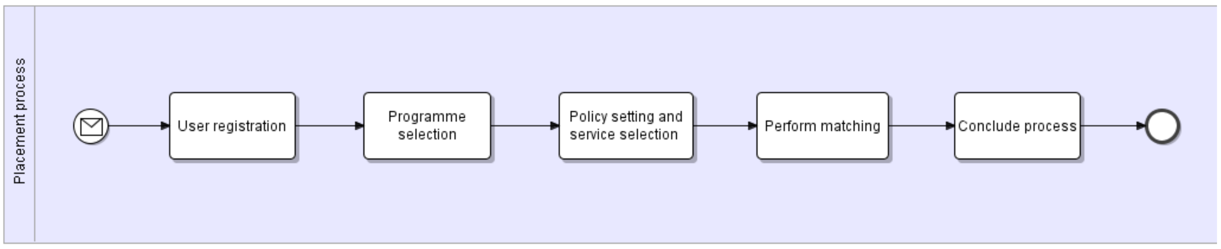


Figure 3.1: Overview of the 2nd Nottingham Student Placement Process

Figure 3.1 gives a high-level overview of the main steps of the Second Nottingham Demonstrator business process on the application layer. On this level the process follows, by and large, the process of the first demonstrator. The improvements are illustrated and explained in the following sections.

3.2 Modelling the Student Placement Process

Figure 3.1 shows a high-level overview of the student placement process. In this section we describe the modelling approach regarding the process model. Section 3.2.1 starts with a description of the business process model on the application level. Section 3.2.2 introduces the security aspects of the process.

3.2.1 Student Placement Process Model

The process is divided into three figures. Figure 3.2 shows the initial phase of the process flow. The process starts with a web service call from the placement administrator website. At the beginning, the process differentiates between learners, who already have a personal data store (PDS), and those, who don't have it. The TAS^3 Discovery Service initially evaluates whether the user has a preferred PDS set for his/her IdP account. If this is the case, the process checks whether the learner really has an account there or not. Learners who don't have an account yet have to go through some registration steps. Other learners do not have to pass these steps. After the learner has provided the required registration data, the placement coordinator has to approve the learner for taking part in the further placement process.

Learners lacking a personal data store have to enter registration data. A PDS is created for them and the registration data is imported into the PDS (Fig. 3.3). The learner receives a notification for the creation. This happens via email and prevents the learner from the need to log-in repeatingly for evaluating whether the coordinator has already performed the approval task.

As opposed to the first demonstrator the coordinator selects the placement programme. The learner gets informed, he/she has to accept the programme-specific terms&conditions.

The business process performs a call the matching service (Figure 3.4). The matching service retrieves the learner's data form his/her PDS, performs a matching, and passes the results back to the learner's PDS. Additionally the results are shown to the learner in the business process. A receipt about data usage and actions performed in the process is generated for the learner and shown to him/her, too. Finally, the process generates and shows also a receipt to the placement coordinator. The step of user giving feedback on the matching service used is skipped in this demonstrator, because this has already been shown in the first Nottingham demonstrator.

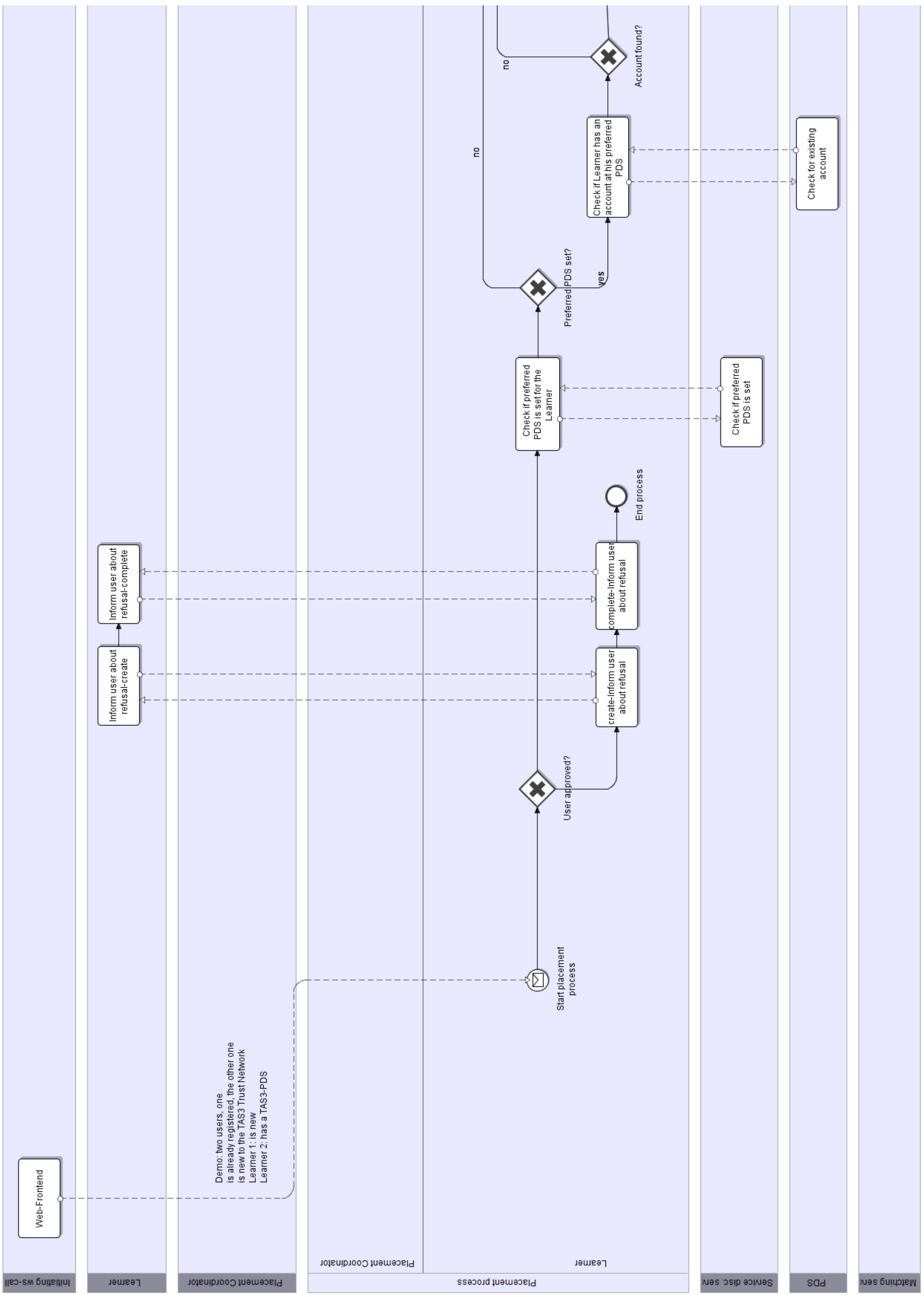

Initiating ws-call
Learner
Placement Coordinator
Placement process
Placement Coordinator
Learner
Service disc. serv.
PDS
Matching serv.
Web-Frontend
Demo: two users, one is already registered, the other one is new to the TAS3 Trust Network
Learner 1: is new
Learner 2: has a TAS3-PDS
Start placement process
User approved?
create-Inform user about refusal
complete-Inform user about refusal
End process
Inform user about refusal-create
Inform user about refusal-complete
Check if preferred PDS is set for the Learner
Check if preferred PDS is set
Preferred PDS set?
yes
no
Check if Learner has an account at his preferred PDS
Check for existing account
Account found?
no

Figure 3.2: 2nd Nottingham Student Placement Process Model – 1

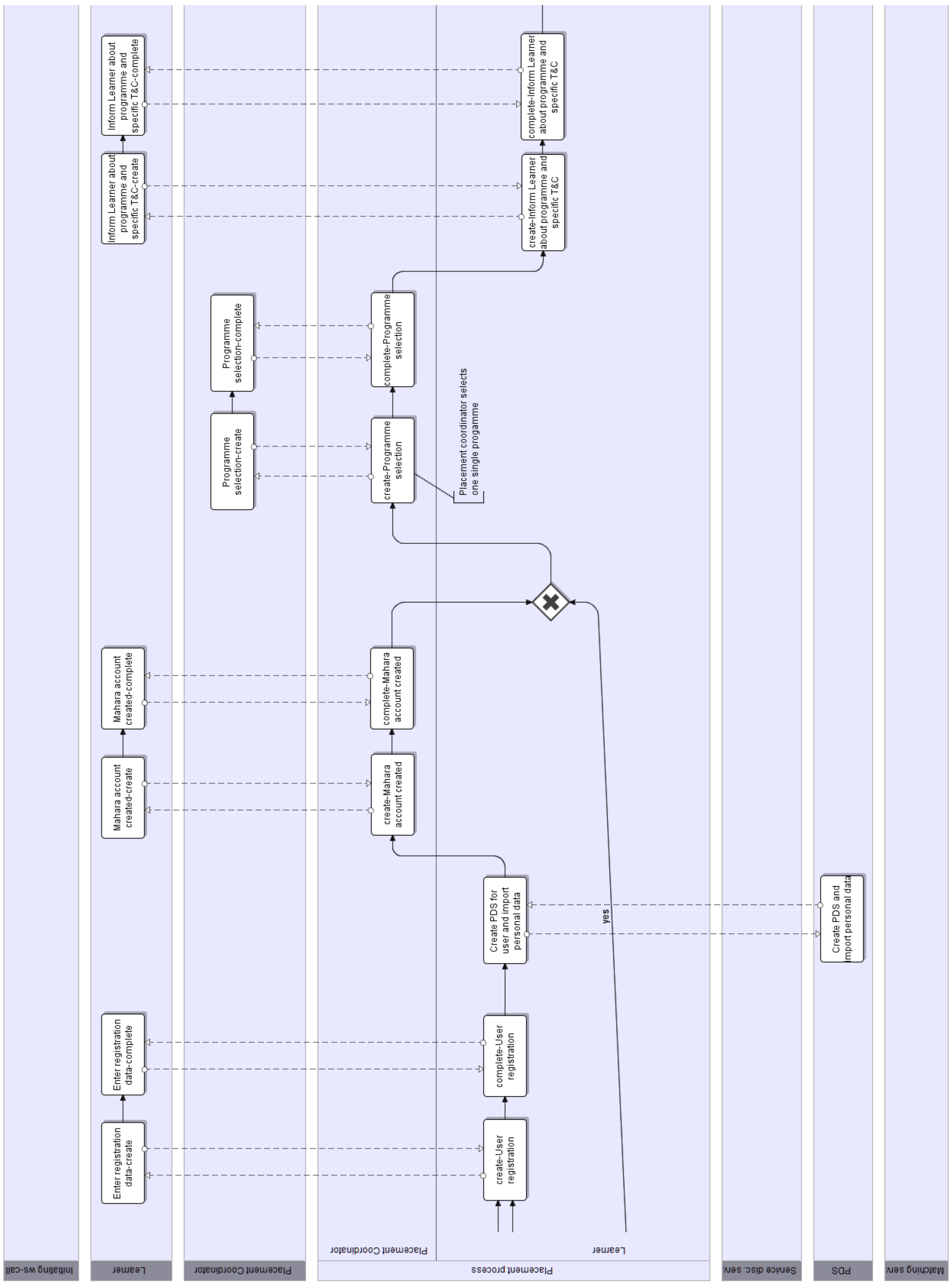

Initiating ws-call
Learner
Placement Coordinator
Placement process
Placement Coordinator
Learner
Service disc. serv.
PDS
Matching serv.
Enter registration data-create
Enter registration data-complete
create-User registration
complete-User registration
Create PDS for user and import personal data
Create PDS and import personal data
yes
Mahara account created-create
Mahara account created-complete
create-Mahara account created
complete-Mahara account created
Programme selection-create
Programme selection-complete
create-Programme selection
complete-Programme selection
Placement coordinator selects one single progamme
Inform Learner about programme and specific T&C-create
Inform Learner about programme and specific T&C-complete
create-Inform Learner about programme and specific T&C
complete-Inform Learner about programme and specific T&C

Figure 3.3: 2nd Nottingham Student Placement Process Model – 2

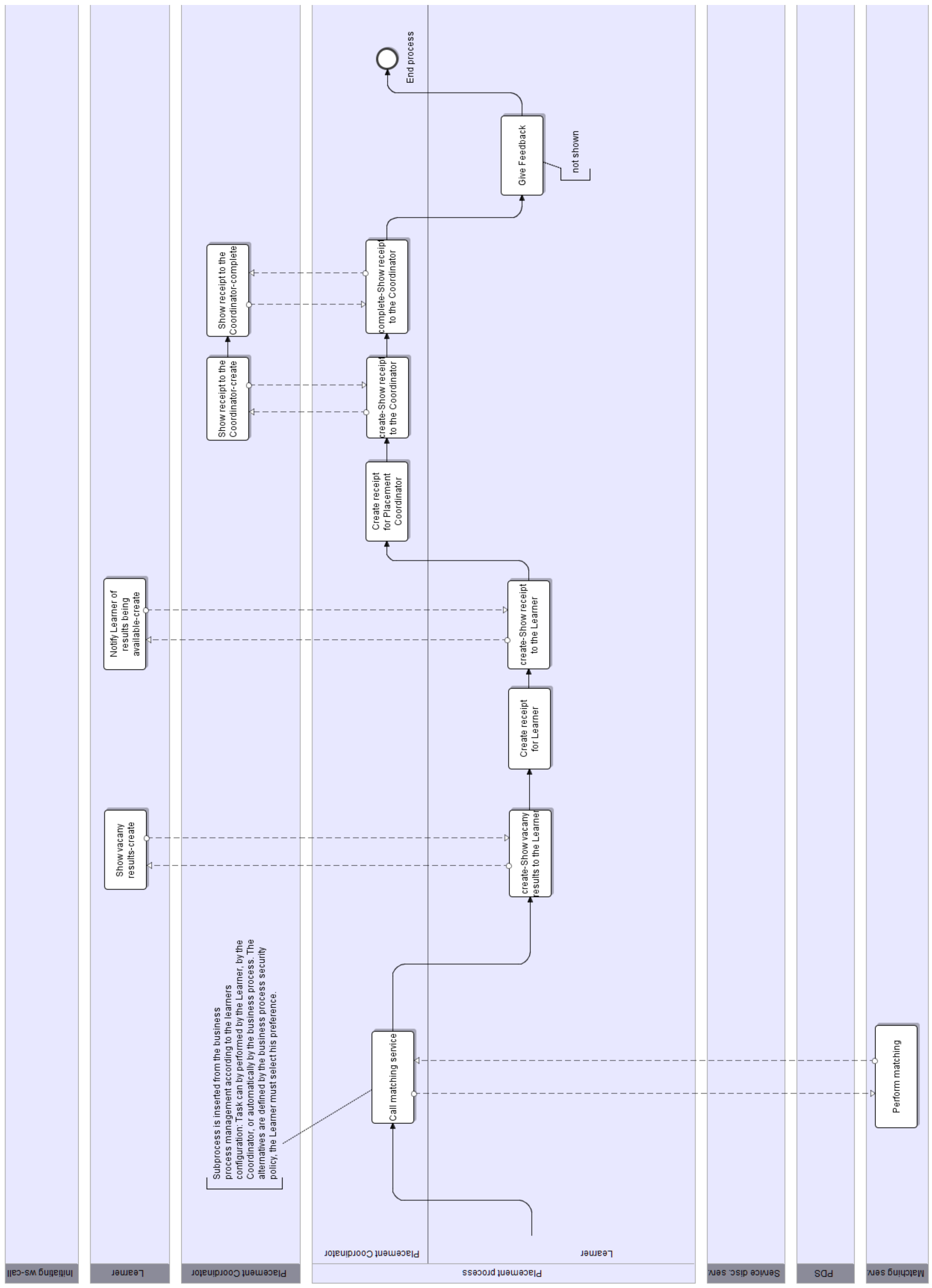

Initiating ws-call
Learner
Placement Coordinator
Placement process
Placement Coordinator
Learner
Service disc. serv.
PDS
Matching serv.
Subprocess is inserted from the business process management according to the learners configuration: Task can by performed by the Learner, by the Coordinator, or automatically by the business process. The alternatives are defined by the business process security policy, the Learner must select his preference.
Call matching service
Perform matching
create-Show vacancy results to the Learner
Show vacany results-create
Create receipt for Learner
create-Show receipt to the Learner
Notify Learner of results being available-create
Create receipt for Placement Coordinator
create-Show receipt to the Coordinator
Show receipt to the Coordinator-create
complete-Show receipt to the Coordinator
Show receipt to the Coordinator-complete
Give Feedback
not shown
End process

Figure 3.4: 2nd Nottingham Student Placement Process Model – 3

3.2.2 Security Annotated Process Model

After modelling the student placement process on the application layer, we add security constraints to the process modell. In [1], Chapter 4.2, we describe our approach to allow specifying security constraints on the level of a business process model with security annotations and give a complete overview about possible annotations and the syntax. In this section, we describe the concrete security annotations of the Second Nottingham Student Placement Process from process start (see Figure 3.2, left side) to process end (see Figure 3.4, right side).

Lane *Placement Coordinator* (Figure 3.5):

≪Assignment: type="Role"≫
says that the lane name "Placement Coordinator" represents also a role name "Placement Coordinator". This implies that all activities in lane "Placement Coordinator" have to executed by a roleholder of the role with the same name.

≪Authn: attribute="(position,contractor), (affiliation,UniNottingham)" idp="idp.tas3.eu"≫ means that all process participants have to be authenticated. Their attributes values "contractor" (as position) and "UniNottingham" (as affiliation) must be asserted by the trusted Identity Provider "idp.tas3.eu".

≪BoD: spec="weak"≫
expresses that within one process instance all activities of lane "Placement Coordinator" have to be performed by the same user. Since the process may take long time, the parameter spec="weak" is used to allow an explicit transfer of responsibility for activities in this lane (a so-called *re-assignment*).

≪SoD: role="Learner"≫
says that a role holder of role "Learner" in this process instance must not be a holder of role "Placement Coordinator" in the same process instance.

Lane *Learner*:

≪Assignment: type="Role"≫
specifies that the lane name "Learner" is also a role with the name "Learner" and that all activities of lane "Learner" have to be performed by role holders of "Learner".

≪Authn: attribute="(position,student), (affiliation,UniNottingham)" idp="idp.tas3.eu"≫
means that all process participants of role "Learner" have to be authenticated. The IdP "idp.tas3.eu" must assure that they hold the attributes values "student" (as position) and "UniNottingham" (as affiliation).

≪BoD≫
specifies that within one process instance all activities of lane "Learner" have to be performed by the same user. A re-assignment is not allowed.

≪SoD: role="Placement Coordinator"≫
declares that a role holder of role "Placement Coordinator" in this process instance must not be a holder of role "Learner" in the same process instance.

Start placement process event:

≪UInvolve:
type="Consent"
target="TAS3 Privacy Terms"
display="yes,no"≫
Here, a user has to give consent to the TAS^3 Privacy Terms. He can select "yes" or "no".

≪UInvolve:
type="Consent"
target="Placement Process T&C"
display="yes,no"≫
Here, a user has to give consent to the placement process terms and conditions. Again, he can select "yes"

or ”no”.

The annotation
≪Audit: policy=”auditAll”≫
enforces a monitoring of all security-relevant events in the business process.

Message flow between the activities *Check if preferred PDS is set for the learner* and *Check if preferred PDS is set*:

The annotation
≪Conf: spec=”encrypted,signed”≫
specifies that the message flow between the business process and the service discovery service will be protected by means of encryption. The messages will be digitally signed to assure the identity of the originators.

Message flow between the activities *Create PDS for user and import personal data* and *Create PDS and import personal data* (Figure 3.6):

The annotation term
≪D-Delegation:
rights=”write”
object=”PDS-of-Learner”
owner=”Learner”
receiver=”PDS”
spec=”grant” ≫
specifies the delegation of data access rights (here: *write*) from the learner to the web service ”PDS” for accessing the learner’s PDS (Personal Data Store) after creating it. This delegation holds for the duration of the activity ”Create PDS for user and import personal data”.

Activity *Call matching service* (Figure 3.7):

≪UInvolve:
type=”SetTrustPolicy”
role=”Learner”
policy=”$TrustPolicy”≫
The learner can specify a trust policy to be used for discovering available and appropriate matching services. The parameter *policy* identifies the trust policy to be used.

≪UInvolve:
type=”SetIAPref”
role=”Learner”
target=”$BPPolicy”
display=”$configurationParameters”≫
The learner is be able to specify, if he/she wants the Placement Coordinator to select a matching service out of the list of suitable ones, or if he/she wants to the select the service by him-/herself. The user interaction implementing this possibility of choice is specified by the ”SetIAPref” annotation.

≪UInvolve:
type=”SelectService”
role=”$BPPolicy/selectServiceRole”
display=”$setOfAvailableServices”≫
This annotation provides the role holder of ”selectServiceRole”, as specified by the user involvement stated above, to select a web service (here: of type *Matchingsservice*) from the set of available web services.

≪D-Delegation:
rights=”read,write”
object=”PDS-of-Learner”

owner="Learner"
receiver="PlacementCoordinator"
spec="grant"
≫
The data access rights *read* and *write* to the learner's PDS are delegated to the role holder of "Placement Coordinator". (This is required because the Placement Coordinator performs the activity *Call matching service*).

Message flow between the activities *Call matching service* and *Perform matching*:

≪D-Delegation:
rights="read,write"
object="PDS-of-Learner"
receiver="Matchingservice"
spec="grant"≫
Specifies the delegation of access rights (read, write) to the learner's PDS to the web service "Matchingservice". (Required because the Matchingservice must access the learner's personal data in his/her PDS to perform the matching. After having performed the matching, the Matchingservice transfers the results to the PDS, so rights for write-access are also required.)

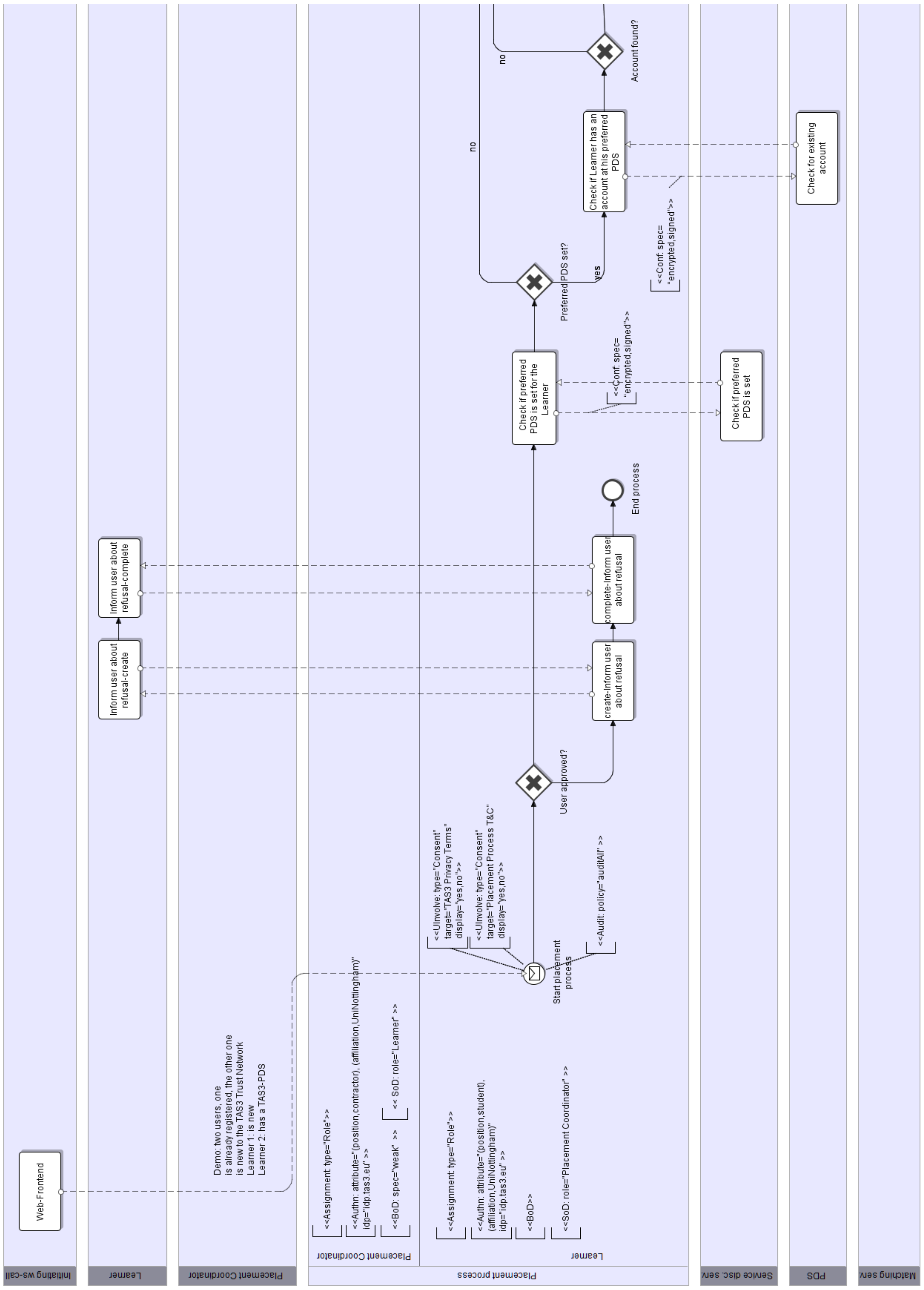

Initiating ws-call
Learner
Placement Coordinator
Placement process
Placement Coordinator
Learner
Service disc. serv.
PDS
Matching serv.
Web-Frontend
Demo: two users, one is already registered, the other one is new to the TAS3 Trust Network
Learner 1: is new
Learner 2: has a TAS3-PDS
<<Assignment: type="Role">>
<<Authn: attribute="(position,contractor), (affiliation,UniNottingham)" idp="idp.tas3.eu" >>
<<BoD: spec="weak" >>
<< SoD: role="Learner" >>
<<Assignment: type="Role">>
<<Authn: attribute="(position,student), (affiliation,UniNottingham)" idp="idp.tas3.eu" >>
<<BoD>>
<<SoD: role="Placement Coordinator" >>
<<UInvolve: type="Consent" target="TAS3 Privacy Terms" display="yes,no">>
<<UInvolve: type="Consent" target="Placement Process T&C" display="yes,no">>
<<Audit: policy="auditAll" >>
Start placement process
User approved?
create-Inform user about refusal
complete-Inform user about refusal
End process
Inform user about refusal-create
Inform user about refusal-complete
Check if preferred PDS is set for the Learner
<<Conf: spec="encrypted,signed">>
Check if preferred PDS is set
Preferred PDS set?
yes
no
Check if Learner has an account at his preferred PDS
<<Conf: spec="encrypted,signed">>
Check for existing account
Account found?
no

Figure 3.5: 2nd Nottingham Student Placement Process with Security and Trust Annotations – 1

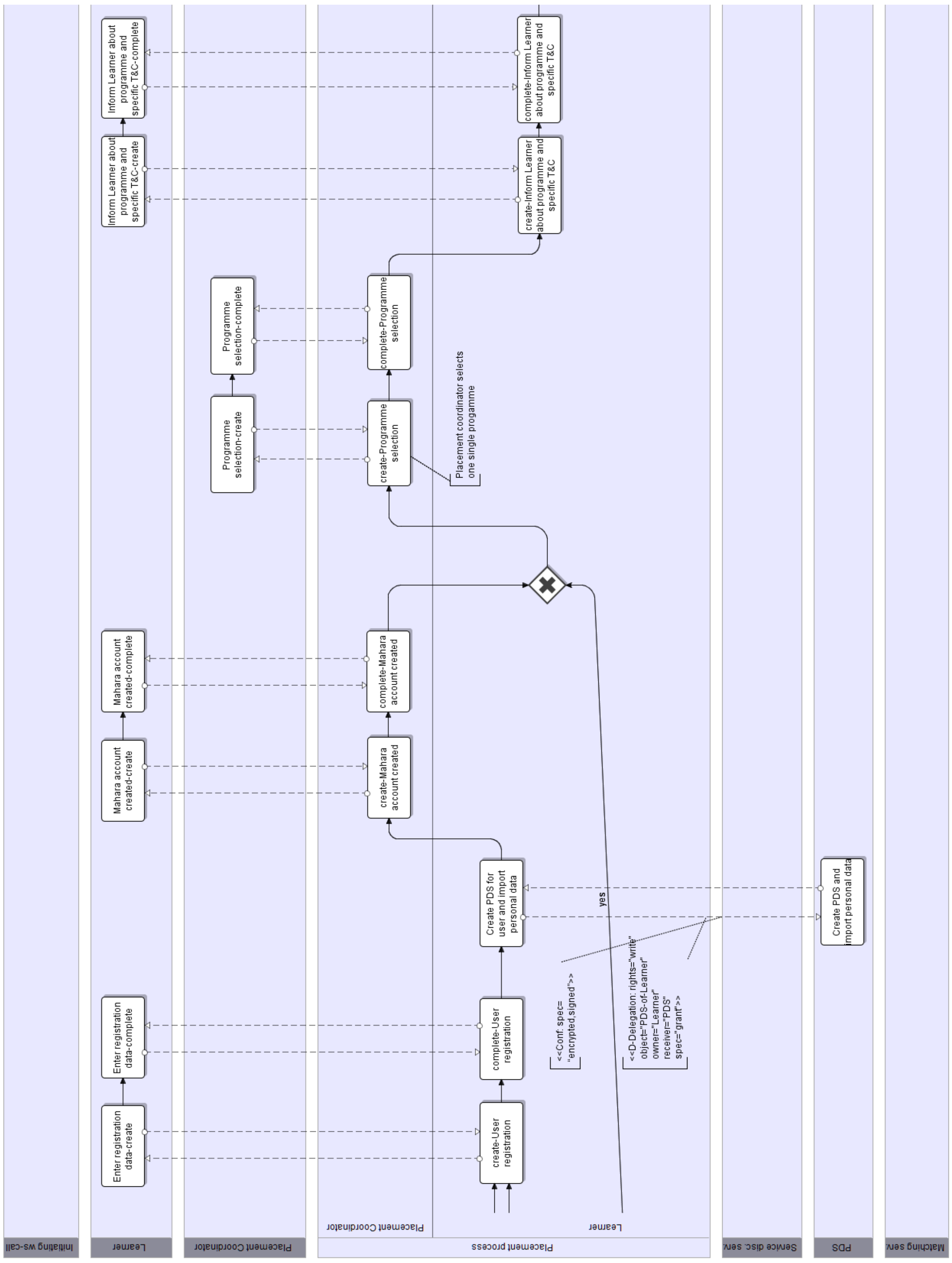

Initiating ws-call
Learner
Placement Coordinator
Placement process
Placement Coordinator
Learner
Service disc. serv.
PDS
Matching serv.
Enter registration data-create
Enter registration data-complete
Mahara account created-create
Mahara account created-complete
Inform Learner about programme and specific T&C-create
Inform Learner about programme and specific T&C-complete
Programme selection-create
Programme selection-complete
create-User registration
complete-User registration
Create PDS for user and import personal data
create-Mahara account created
complete-Mahara account created
create-Programme selection
complete-Programme selection
create-Inform Learner about programme and specific T&C
complete-Inform Learner about programme and specific T&C
Placement coordinator selects one single progamme
yes
<<Conf: spec="encrypted,signed">>
<<D-Delegation: rights="write" object="PDS-of-Learner" owner="Learner" receiver="PDS" spec="grant">>
Create PDS and import personal data

Figure 3.6: 2nd Nottingham Student Placement Process with Security and Trust Annotations – 2

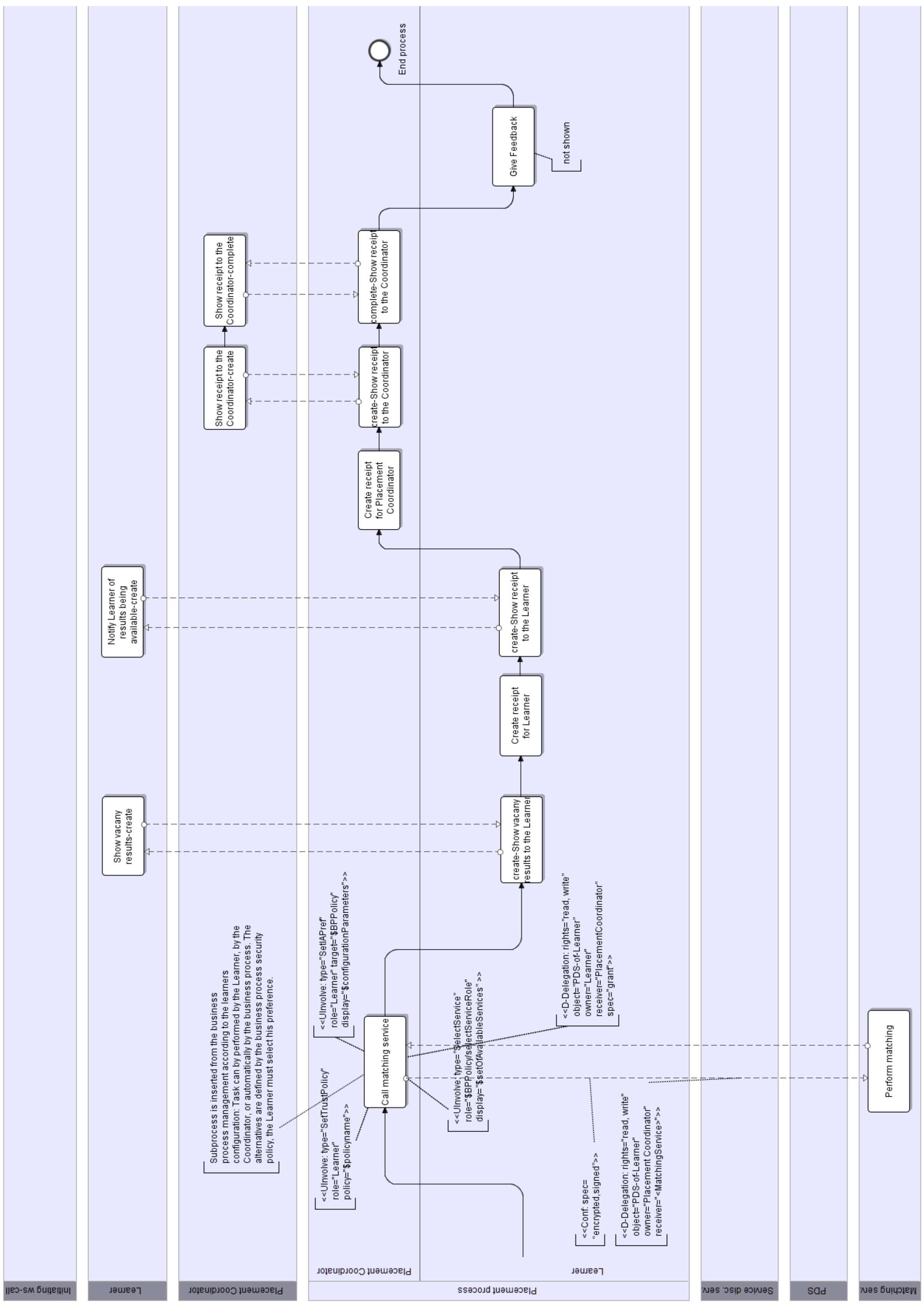

Initiating ws-call
Learner
Placement Coordinator
Placement process
Placement Coordinator
Learner
Service disc. serv.
PDS
Matching serv.
Show vacany results-create
Notify Learner of results being available-create
Subprocess is inserted from the business process management according to the learners configuration; Task can by performed by the Learner, by the Coordinator, or automatically by the business process. The alternatives are defined by the business process security policy, the Learner must select his preference.
Show receipt to the Coordinator-create
Show receipt to the Coordinator-complete
<<UInvolve: type="SetTrustPolicy" role="Learner" policy="$policyname">>
<<UInvolve: type="SetAPref" role="Learner" target="$BPPolicy" display="$configurationParameters">>
Call matching service
Create receipt for Placement Coordinator
create-Show receipt to the Coordinator
complete-Show receipt to the Coordinator
End process
<<UInvolve: type="SelectService" role="$BPPolicy/selectServiceRole" display="$setOfAvailableServices">>
<<Conf: spec="encrypted,signed">>
<<D-Delegation: rights="read, write" object="PDS-of-Learner" owner="Learner" receiver="PlacementCoordinator" spec="grant">>
<<D-Delegation: rights="read, write" object="PDS-of-Learner" owner="Placement Coordinator" receiver="<MatchingService>">>
create-Show vacany results to the Learner
Create receipt for Learner
create-Show receipt to the Learner
Give Feedback
not shown
Perform matching

Figure 3.7: 2nd Nottingham Student Placement Process with Security and Trust Annotations – 3

3.3 Operational sequence and involved components

In the preceding section we described the process model in detail. This section focuses on the integration with other components of the TAS^3 framework. The status of the implementation of the components is described in the second iteration of the WP3 Implementation Deliverable, D3.2.2 ([4]).

Workpackage 3 comprises the development of the following components:

- Business Process Security Modelling and Configuration Component (T3-BP-SMC)
- Apache ODE Business Process Execution Engine (T3-BP-ENGINE-ODE)
- Business Process User Client (T3-BP-CLIENT)
- Business Process Policy Enforcement Point (T3-PEP-BP)
- Business Process Policy Decision Point (T3-PDP-BP)
- Business Process Policy Information Point (T3-BP-PIP)
- Business Process Delegation Service (T3-BP-DEL)

The components are described in detail in Deliverable D3.2 [4].

The demonstrator uses all of these components. In the modelling phase the process designer creates the process model using the Intalio Designer business process modelling tool (T3-EXT-INTALIO-DESIGNER). The Business Process Security Modelling and Configuration Component (T3-BP-SMC) extracts security annotations from the processs model in the next step. Depending on the annotation types the SMC transforms the annotations into business process policy elements or adapts the process model according to the security requirements specified by the annotations.

At runtime the Business Process User Client component (T3-BP-CLIENT) controls all aspects of interaction of subjects with the business process: authentication, role management, task assignment etc. The CLIENT comprises also a user interface for human interaction with the business process. The BP Policy Enforcement Point, the BP Policy Decision Point and the BP Policy Information Point interact tightly to enforce the security constraints of the demonstrator use case. The Delegation Service is the business-process-specific component of the T3-DEL component.

The business process of the Second Nottingham Demonstrator not only involves the business process specific TAS^3 components, but also other components of the TAS^3 framework, developed by our project partners. These are:

- T3-DEL: Delegation Service, developed by KENT
- T3-LOG-SAWS: Secure Auditing Web Service, developed by KENT
- T3-PDP-M: Policy Decision Point/Master PDP, developed by KENT
- T3-PDP-P: PERMIS PDP, developed by KENT
- T3-PDP-T: Trust Policy Decision Point, developed by TUE
- T3-SP-MATCHER: Job Profile Matching Service, developed by NOT
- T3-ZXID-LINUX-X86: ZXID library and tools for TAS3 in Apache, Java, PHP, and C, developed by RIS
- T3-ZXID-SRC: Sources for the ZXID package, developed by RIS

The following sequence diagram gives an overview of the component interactions in the Second Nottingham Demonstrator (Figures 3.8 to 3.10).

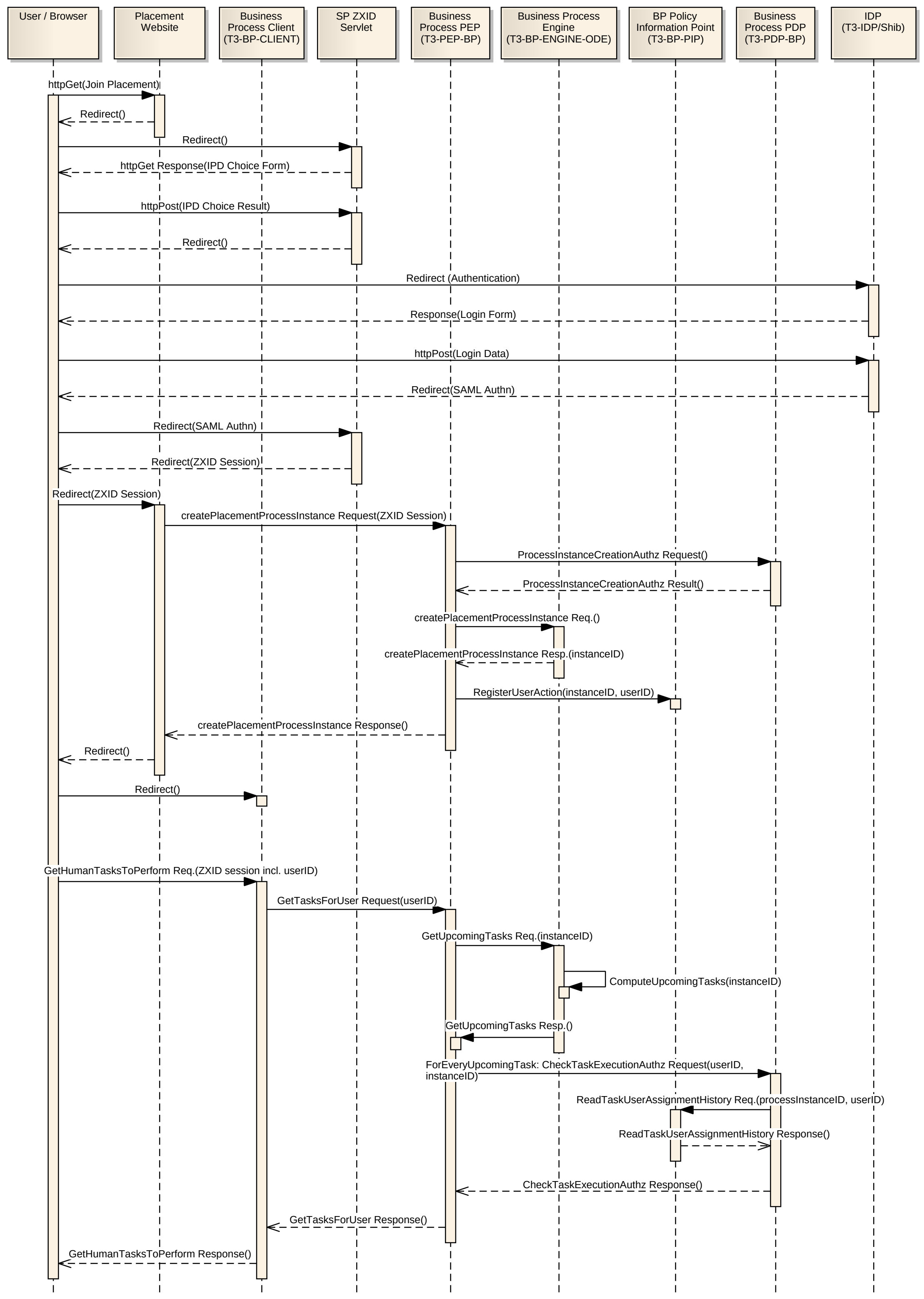

User / Browser
Placement Website
Business Process Client (T3-BP-CLIENT)
SP ZXID Servlet
Business Process PEP (T3-PEP-BP)
Business Process Engine (T3-BP-ENGINE-ODE)
BP Policy Information Point (T3-BP-PIP)
Business Process PDP (T3-PDP-BP)
IDP (T3-IDP/Shib)
httpGet(Join Placement)
Redirect()
Redirect()
httpGet Response(IPD Choice Form)
httpPost(IPD Choice Result)
Redirect()
Redirect (Authentication)
Response(Login Form)
httpPost(Login Data)
Redirect(SAML Authn)
Redirect(SAML Authn)
Redirect(ZXID Session)
Redirect(ZXID Session)
createPlacementProcessInstance Request(ZXID Session)
ProcessInstanceCreationAuthz Request()
ProcessInstanceCreationAuthz Result()
createPlacementProcessInstance Req.()
createPlacementProcessInstance Resp.(instanceID)
RegisterUserAction(instanceID, userID)
createPlacementProcessInstance Response()
Redirect()
Redirect()
GetHumanTasksToPerform Req.(ZXID session incl. userID)
GetTasksForUser Request(userID)
GetUpcomingTasks Req.(instanceID)
ComputeUpcomingTasks(instanceID)
GetUpcomingTasks Resp.()
ForEveryUpcomingTask: CheckTaskExecutionAuthz Request(userID, instanceID)
ReadTaskUserAssignmentHistory Req.(processInstanceID, userID)
ReadTaskUserAssignmentHistory Response()
CheckTaskExecutionAuthz Response()
GetTasksForUser Response()
GetHumanTasksToPerform Response()

Figure 3.8: Nottingham employability demonstrator sequence diagram – 1

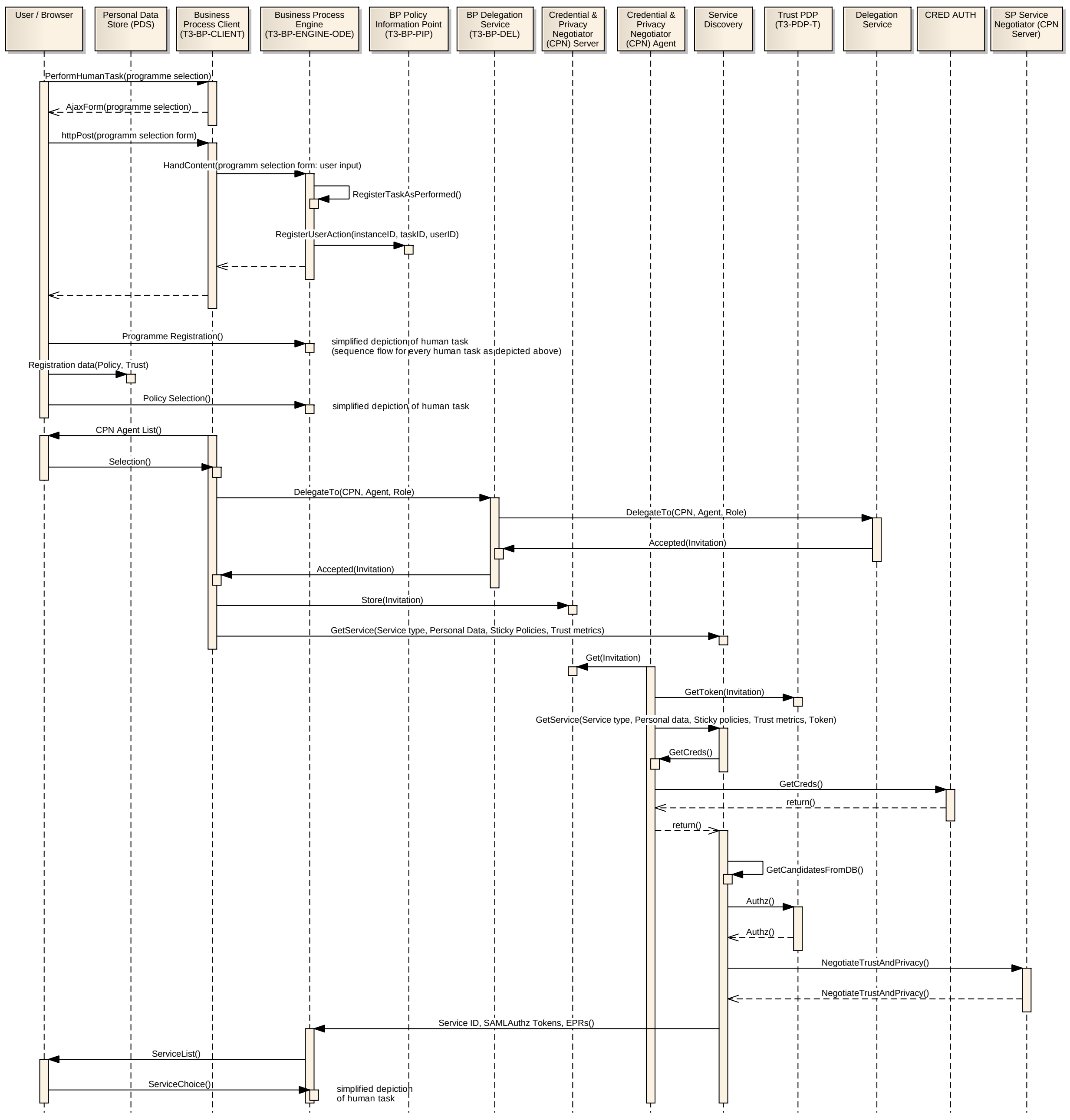

User / Browser
Personal Data Store (PDS)
Business Process Client (T3-BP-CLIENT)
Business Process Engine (T3-BP-ENGINE-ODE)
BP Policy Information Point (T3-BP-PIP)
BP Delegation Service (T3-BP-DEL)
Credential & Privacy Negotiator (CPN) Server
Credential & Privacy Negotiator (CPN) Agent
Service Discovery
Trust PDP (T3-PDP-T)
Delegation Service
CRED AUTH
SP Service Negotiator (CPN Server)
PerformHumanTask(programme selection)
AjaxForm(programme selection)
httpPost(programm selection form)
HandContent(programm selection form: user input)
RegisterTaskAsPerformed()
RegisterUserAction(instanceID, taskID, userID)
Programme Registration()
simplified depiction of human task
(sequence flow for every human task as depicted above)
Registration data(Policy, Trust)
Policy Selection()
simplified depiction of human task
CPN Agent List()
Selection()
DelegateTo(CPN, Agent, Role)
DelegateTo(CPN, Agent, Role)
Accepted(Invitation)
Accepted(Invitation)
Store(Invitation)
GetService(Service type, Personal Data, Sticky Policies, Trust metrics)
Get(Invitation)
GetToken(Invitation)
GetService(Service type, Personal data, Sticky policies, Trust metrics, Token)
GetCreds()
GetCreds()
return()
return()
GetCandidatesFromDB()
Authz()
Authz()
NegotiateTrustAndPrivacy()
NegotiateTrustAndPrivacy()
Service ID, SAMLAuthz Tokens, EPRs()
ServiceList()
ServiceChoice()
simplified depiction
of human task

Figure 3.9: Nottingham employability demonstrator sequence diagram – 2

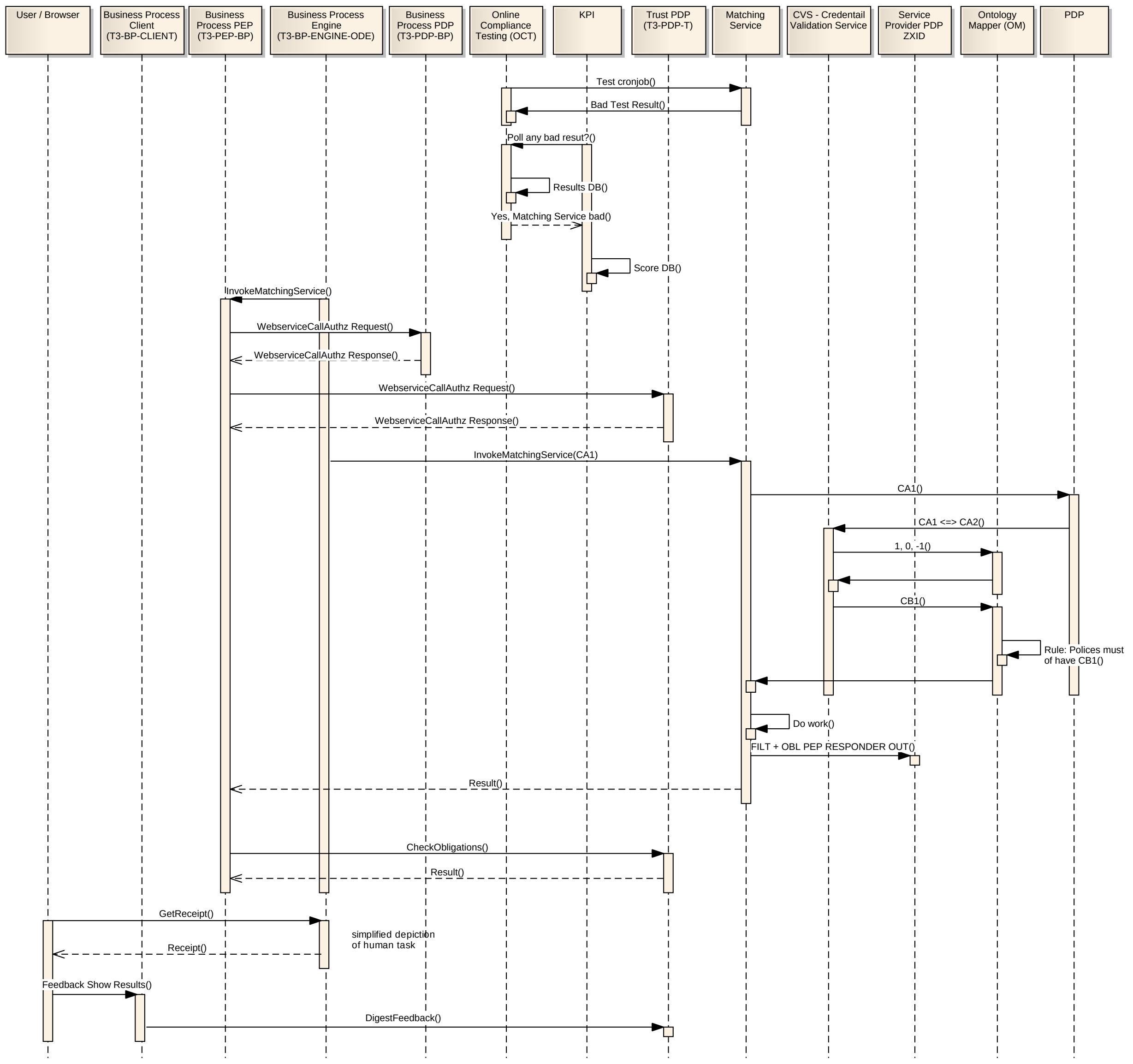

User / Browser
Business Process Client (T3-BP-CLIENT)
Business Process PEP (T3-PEP-BP)
Business Process Engine (T3-BP-ENGINE-ODE)
Business Process PDP (T3-PDP-BP)
Online Compliance Testing (OCT)
KPI
Trust PDP (T3-PDP-T)
Matching Service
CVS - Credentail Validation Service
Service Provider PDP ZXID
Ontology Mapper (OM)
PDP
Test cronjob()
Bad Test Result()
Poll any bad resut?()
Results DB()
Yes, Matching Service bad()
Score DB()
InvokeMatchingService()
WebserviceCallAuthz Request()
WebserviceCallAuthz Response()
WebserviceCallAuthz Request()
WebserviceCallAuthz Response()
InvokeMatchingService(CA1)
CA1()
CA1 <=> CA2()
1, 0, -1()
CB1()
Rule: Polices must of have CB1()
Do work()
FILT + OBL PEP RESPONDER OUT()
Result()
CheckObligations()
Result()
GetReceipt()
Receipt()
simplified depiction of human task
Feedback Show Results()
DigestFeedback()

Figure 3.10: Nottingham employability demonstrator sequence diagram – 3

3.4 Implementation

In this section the implementation of the second Nottingham demonstrator is described in more detail. While in section 2.4 we described the implementation of the first Nottingham demonstrator as a whole, for the second demonstrator we focus on the innovations.

3.4.1 Startup phase

The startup phase consists of the following steps (see Figure 3.11):

1. Single sign-on
2. Acceptance of terms&conditions

For modelling the security facets of the second Nottingham business process we used annotations specifying who is allowed to perform which tasks (see section 3.2.1). To enforce these security constraints we use two mechanisms:
1. When a user logs on to the BPMS using SSO we check his/her attributes. Only users having the attributes specified in one of the process lanes (*Learner* and *Placement Coordinator*) can execute tasks. Checking the attributes is implemented in the PEP by using ZXID functionality.
2. We enforce the user-task assignment constraints specified in form of annotations (*Assignment*, *BoD*, *SoD*). Due to Intalio|Designer software architecture matters and limited ressources we had to implement this using proprietary Intalio functionality instead of having a proper solution of constraint transformation to the enforcement level. (See section 6.2.2.2 of the evaluation for further details.)

One of the innovations of the second demonstrator is the automatic transformation of *User Involvement* security annotations into process model components. For instance the annotation <<*UInvolve: type="Consent" target="TAS3 Privacy Terms" display="yes,no"*>> gets transformed into a subprocess fragment, which requests the learner to accept the TAS^3 terms and conditions, and handles a possible decline. This transformation is part of the functionality of the Security Modelling and Configuration Component (T3-BP-SMC).

3.4.2 Registration phase

The following steps take place in the registration phase:

1. Check for existing PDS
2. User registration
3. Creation of a PDS for the user and import of registration data

Steps 2 and 3 are performed only if the learner does not yet have a PDS.

Existing TAS^3 technology is applied to execute the first three steps mentioned above. The ZXID data discovery functionality allows to detemine if a PDS for learner already exists. If the user has no PDS up to now, he must register for the placement process. To reduce the effort for the user the registration is assisted by several calls to the IdP. These calls request personal data of the user already being available at the IdP, and allow a reuse within the scope of the business process. Accepting the placement process terms&conditions the user has already permitted these calls.

For creating a PDS, Nottingham has enhanced the e-portfolio system Mahara[1] with interfaces for automatic account creation and data transfer based on the Leap2A[2] specification. The business process triggers the creation of an PDS account for the user and passes the registration data on to the PDS. By default, access to the PDS is granted only to the learner. Later in the placement process the matching service requires

[1]Mahara website: http://mahara.org/
[2]Leap2A website: http://www.leapspecs.org/2A/index.php

access to the earner's PDS to perform the matching and write the results back to the PDS. To allow the learner to delegate access for his/her PDS to the matching service the learner retrieves a bearer token in form of a secret URL from the PDS. This URL points to a view of the PDS, granting specified access rights to selected data elements in the PDS.

Another possibility of realizing delegation is shown when the call to the matching service is executed with the credentials of the learner. Therefor the learner token is delegated to the placement coordinator using the Delegation Issuing Service. The placement coordinator is able to decide whether he wants to accept the delegation or not.

Figures 3.12 and 3.13 depict the component interactions during registration phase.

3.4.3 Programme selection phase

The following steps take place in the programme selection phase:

1. Selection of a placement programme
2. Acceptance of placement programme terms&conditions
3. Adjust PDS security policies
4. Setting/adjustment of security policies for the PDS

Figures 3.13 and 3.14 depict the component interactions during programme selection phase.

The placement coordinator (PC) selects a placement programme he considers to be adequate for the learner. The personal data of the learner serves as a basis for this assessment. A request to the Mahara PDP is sent to evaluate whether the PC is allowed to see the data (*Check data access rights on Learner data*). In the demonstrator the learner has accepted the Placement Process T&C including a statement, saying that the default policy for a newly created PDS contains an entry that grants data access rights to the placement coordinator being involved in the specific business process instance by default. (For the demonstrator we didn't model all possible applications flows, e.g. for the case that the learner has changed the PDS policy in the meantime, so that the coordinator can't access the data and therefore the learner must reset his policy to give data access to the PC, etc. We supposed the default policy to be still valid.)

After the placement coordinator having selected an appropriate programme the learner has to accept the programme terms&conditions to proceed with the process.

Before discovering and selecting a matching service the learner is asked to set security and trust policies. The security policy setting concerns the access rights to the learners personal data and is done within Mahara, where the personal data is stored. The learner is forwarded to his/her personal data store by a business process user form.

3.4.4 Service selection and matching service call phase

The service selection and matching service call phase comprises the following steps:

1. Selection of a trust policy for the placement programme
2. Business process policy configuration
3. Matching service discovery and selection
4. Matching service call
5. Import results into PDS

Setting a trust policy regarding the matching services to be discovered is done using a busines process interface. The learner can select a minimal reputation and a minimal reliability score he claims the matching service to fulfill.

In addition to the functionality of the first demonstrator, not the learner but the placement coordinator may be responsible for selecting the matching service to be called. This depends on the learner's preferences specified during the user task *Business process policy configuration*.

The task *Service discovery* sends a service discovery request using ZXID functionality. Parameters used for discovery are, amongst others, the servicetype and both types of trust policies. The discovery service queries whether appropriate services are available. If no services matching the learner's demands can be found, the user can change his/her preferences and trigger discovery again.

When at least one service has been found one of the available services must be selected to be used in the application. Depending on the preferences the learner has set before the selection is performed either by the learner or by the placement coordinator. The selected service is stored in the PIP as process instance-specific information. Relying on this entry the PEP executes the call to the matching service, triggered by the business process. Just before performing the call itself, the PEP re-checks the validity of the service against the policies specified by the learner. This is inevitable because the service rating may have changed in the meantime. If the service no longer satisfies the demands of the learner, he/she has to select a different service.

Figures 3.15 and 3.17 show the component interactions during service selection and matching service call phase and the conclusion process phase.

3.4.5 Conclusion process phase

The Process conclusion phase (Figure 3.15) includes the following steps:

1. Create receipt for the learner
2. Create receipt for the placement coordinator

For creating helpful and informative receipts all significant events are logged to the Audit Bus (T3-AUTH-BUS) during process execution. The Dashboard provides functionality to present the logs to the end-users. Additionally summarized and personalised forms of the audit log are displayed to all involved process participants, to the learner as well as to the placement coordinator. (See Figure 3.18.)

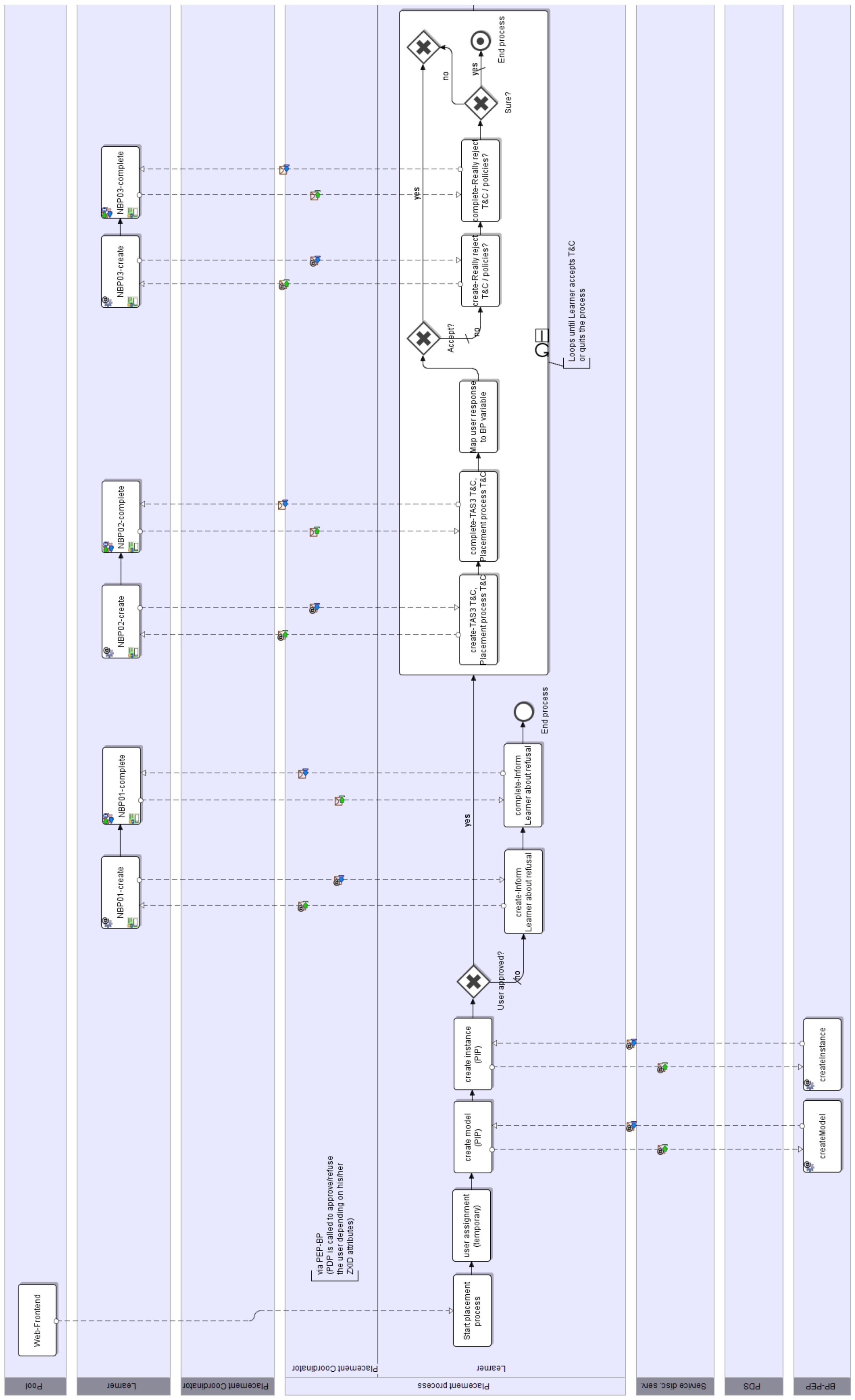

Pool
Web-Frontend
Learner
NBP01-create
NBP01-complete
NBP02-create
NBP02-complete
NBP03-create
NBP03-complete
Placement Coordinator
Placement process
Placement Coordinator
Learner
via PEP-BP (PDP is called to approve/refuse the user depending on his/her ZXID attributes)
Start placement process
user assignment (temporary)
create model (PIP)
create instance (PIP)
User approved?
no
yes
create-Inform Learner about refusal
complete-Inform Learner about refusal
End process
create-TAS3 T&C, Placement process T&C
complete-TAS3 T&C, Placement process T&C
Map user response to BP variable
Accept?
no
yes
create-Really reject T&C / policies?
complete-Really reject T&C / policies?
Sure?
yes
no
End process
Loops until Learner accepts T&C or quits the process
Service disc. serv.
PDS
BP-PEP
createModel
createInstance

Figure 3.11: Nottingham Student Placement Process Model – 1

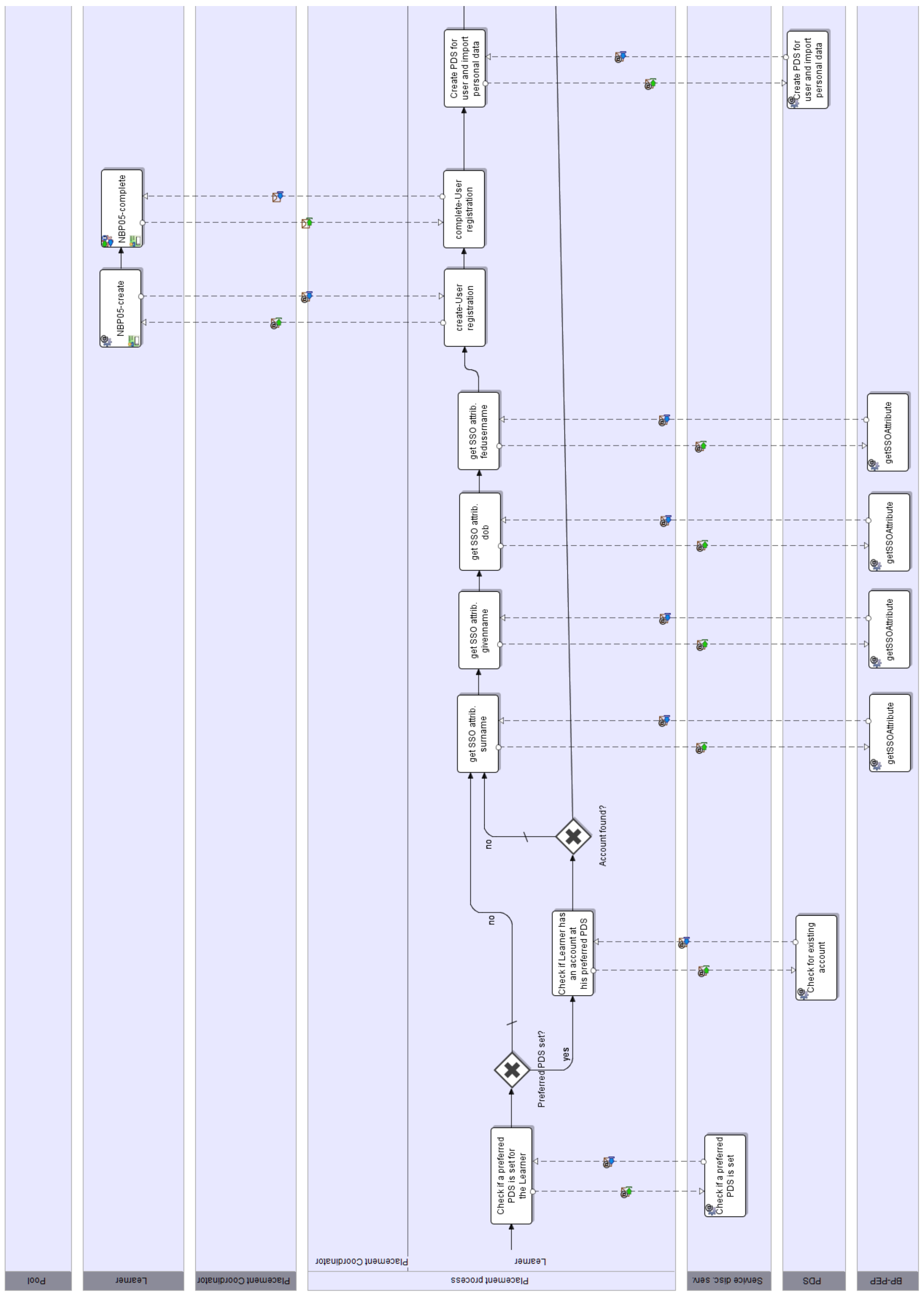

Pool
Learner
Placement Coordinator
Placement process
Placement Coordinator
Learner
Service disc. serv.
PDS
BP-PEP
Check if a preferred PDS is set for the Learner
Check if a preferred PDS is set
Preferred PDS set?
yes
no
Check if Learner has an account at his preferred PDS
Check for existing account
Account found?
no
get SSO attrib. surname
get SSO attrib. givenname
get SSO attrib. dob
get SSO attrib. fedusername
getSSOAttribute
getSSOAttribute
getSSOAttribute
getSSOAttribute
create-User registration
complete-User registration
NBP05-create
NBP05-complete
Create PDS for user and import personal data
Create PDS for user and import personal data

Figure 3.12: Nottingham Student Placement Process Model – 2

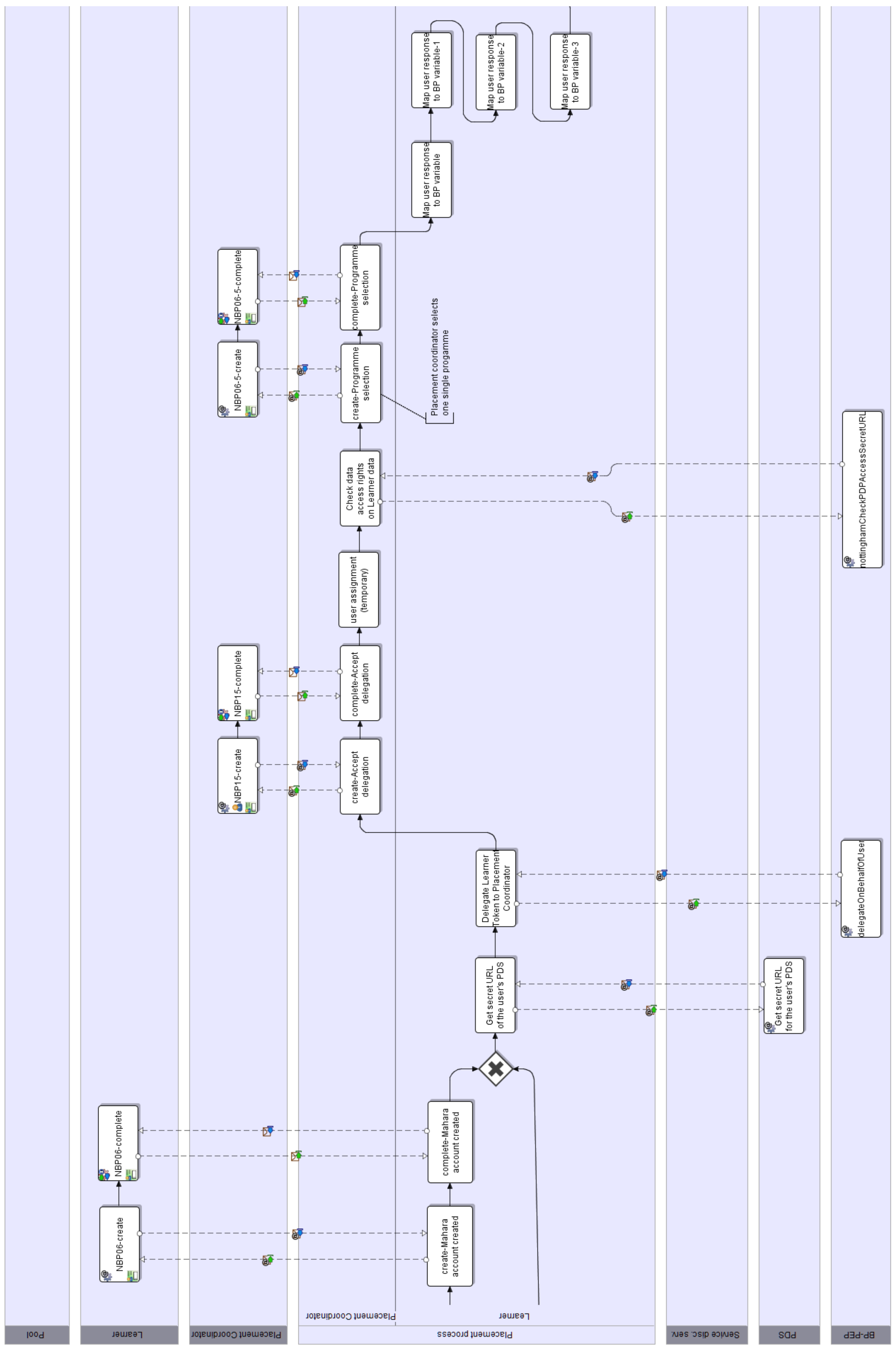

Pool
Learner
Placement Coordinator
Placement process
Placement Coordinator
Learner
Service disc. serv.
PDS
BP-PEP
NBP06-create
NBP06-complete
create-Mahara account created
complete-Mahara account created
Get secret URL of the user's PDS
Get secret URL for the user's PDS
Delegate Learner Token to Placement Coordinator
delegateOnBehalfOfUser
create-Accept delegation
complete-Accept delegation
NBP15-create
NBP15-complete
user assignment (temporary)
Check data access rights on Learner data
nottinghamCheckPDPAccessSecretURL
create-Programme selection
complete-Programme selection
NBP06-5-create
NBP06-5-complete
Placement coordinator selects one single progamme
Map user response to BP variable
Map user response to BP variable-1
Map user response to BP variable-2
Map user response to BP variable-3

Figure 3.13: Nottingham Student Placement Process Model – 3

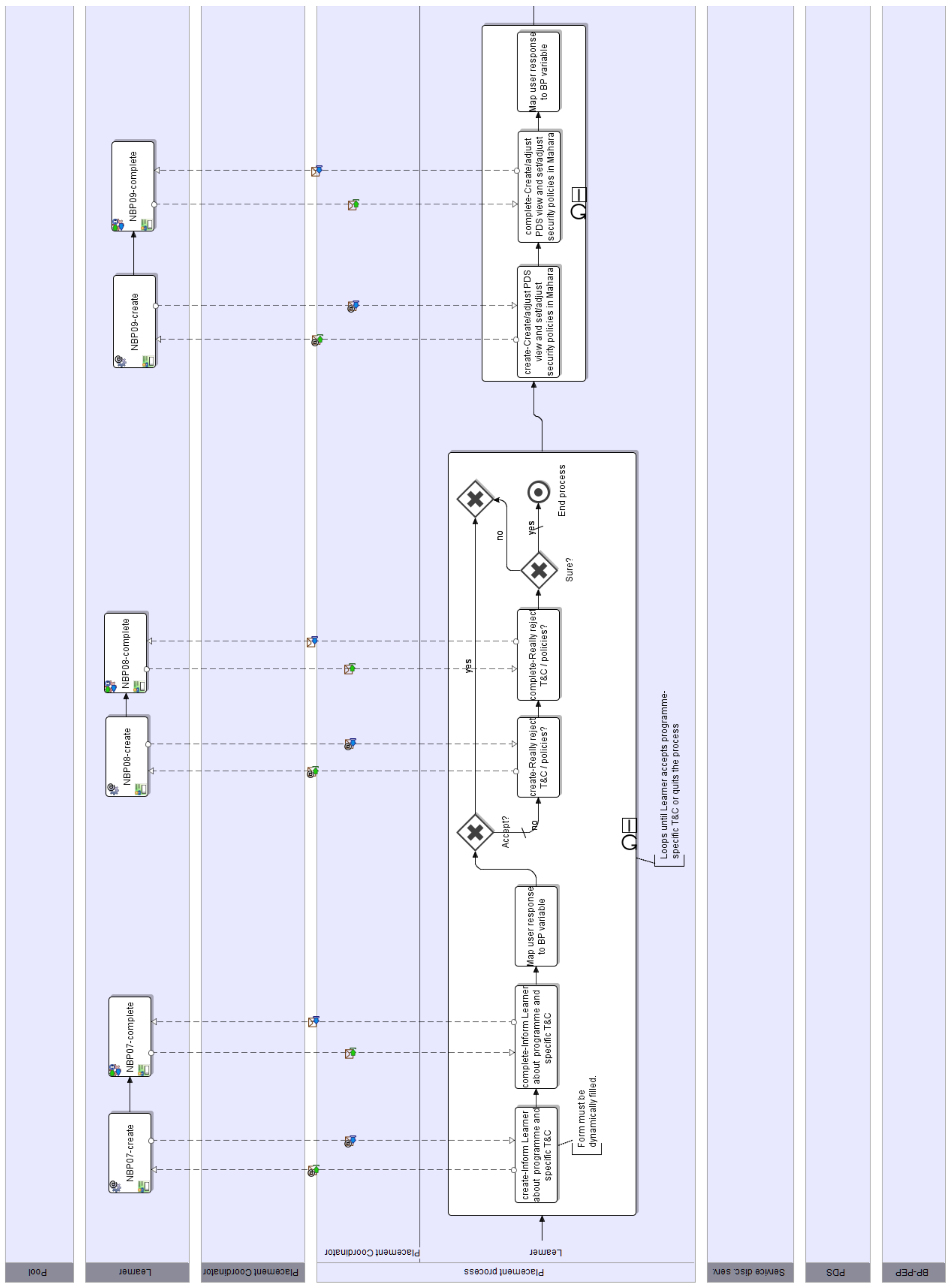

Pool
Learner
Placement Coordinator
Placement process
Learner
Placement Coordinator
Service disc. serv.
PDS
BP-PEP
NBP07-create
NBP07-complete
NBP08-create
NBP08-complete
NBP09-create
NBP09-complete
create-Inform Learner about programme and specific T&C
Form must be dynamically filled.
complete-Inform Learner about programme and specific T&C
Map user response to BP variable
Accept?
no
yes
create-Really reject T&C / policies?
complete-Really reject T&C / policies?
Sure?
yes
no
End process
Loops until Learner accepts programme-specific T&C or quits the process
create-Create/adjust PDS view and set/adjust security policies in Mahara
complete-Create/adjust PDS view and set/adjust security policies in Mahara
Map user response to BP variable

Figure 3.14: Nottingham Student Placement Process Model – 4

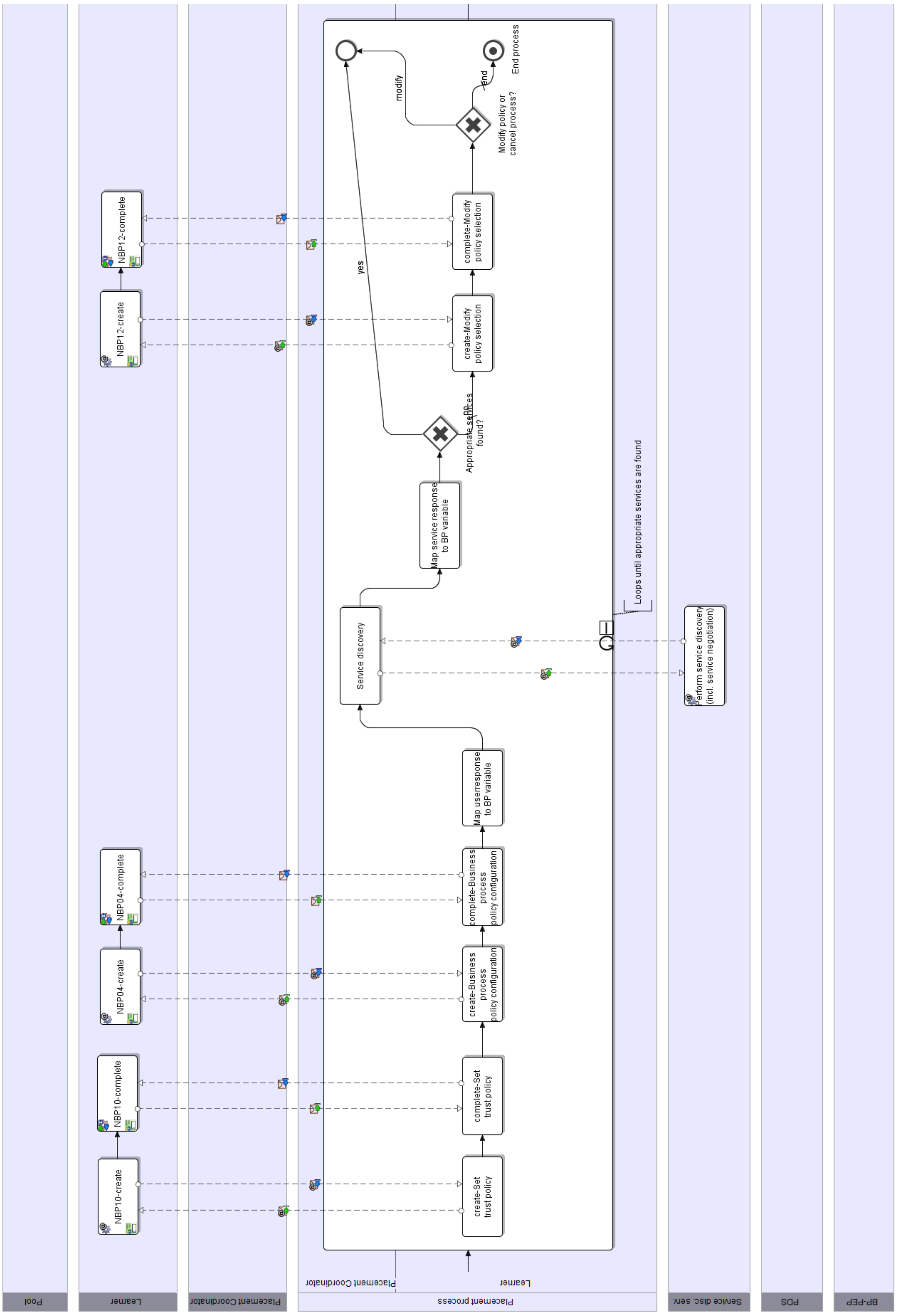

Pool
Learner
Placement Coordinator
Placement process
Service disc. serv.
PDS
BP-PEP
Placement Coordinator
Learner
NBP10-create
NBP10-complete
NBP04-create
NBP04-complete
NBP12-create
NBP12-complete
create-Set trust policy
complete-Set trust policy
create-Business process policy configuration
complete-Business process policy configuration
Map userresponse to BP variable
Service discovery
Map service response to BP variable
Appropriate services found?
yes
create-Modify policy selection
complete-Modify policy selection
Modify policy or cancel process?
modify
End
End process
Loops until appropriate services are found
Perform service discovery (incl. service negotiation)

Figure 3.15: Nottingham Student Placement Process Model – 5

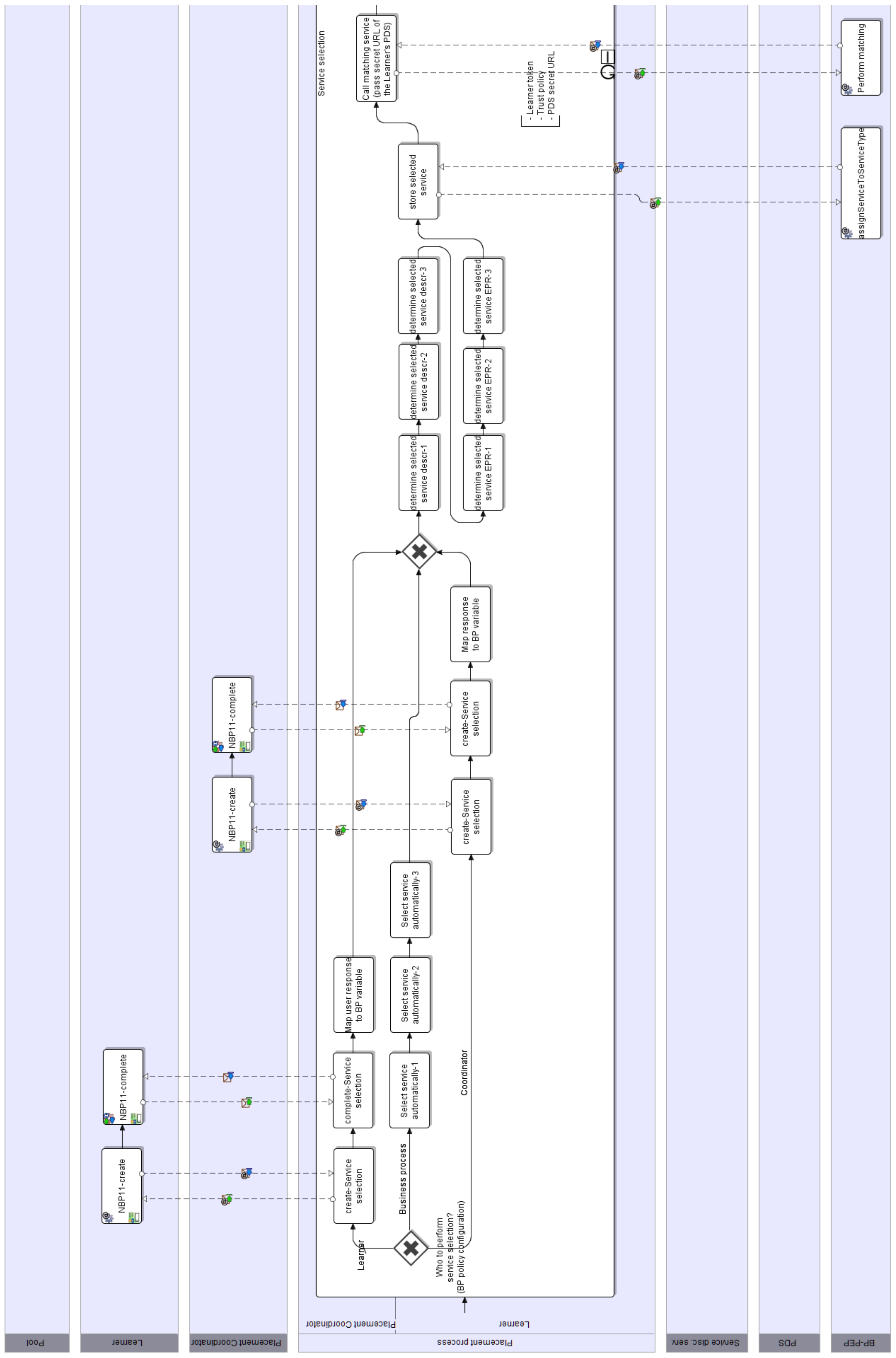

Pool
Learner
Placement Coordinator
Placement process
Service disc. serv.
PDS
BP-PEP
Placement Coordinator
Learner
Service selection
Learner
Business process
Coordinator
Who to perform service selection? (BP policy configuration)
create-Service selection
complete-Service selection
Map user response to BP variable
Select service automatically-1
Select service automatically-2
Select service automatically-3
create-Service selection
create-Service selection
Map response to BP variable
NBP11-create
NBP11-complete
NBP11-create
NBP11-complete
determine selected service descr-1
determine selected service descr-2
determine selected service descr-3
determine selected service EPR-1
determine selected service EPR-2
determine selected service EPR-3
store selected service
Call matching service (pass secret URL of the Learner's PDS)
- Learner token
- Trust policy
- PDS secret URL
Perform matching
assignServiceToServiceType

Figure 3.16: Nottingham Student Placement Process Model – 6

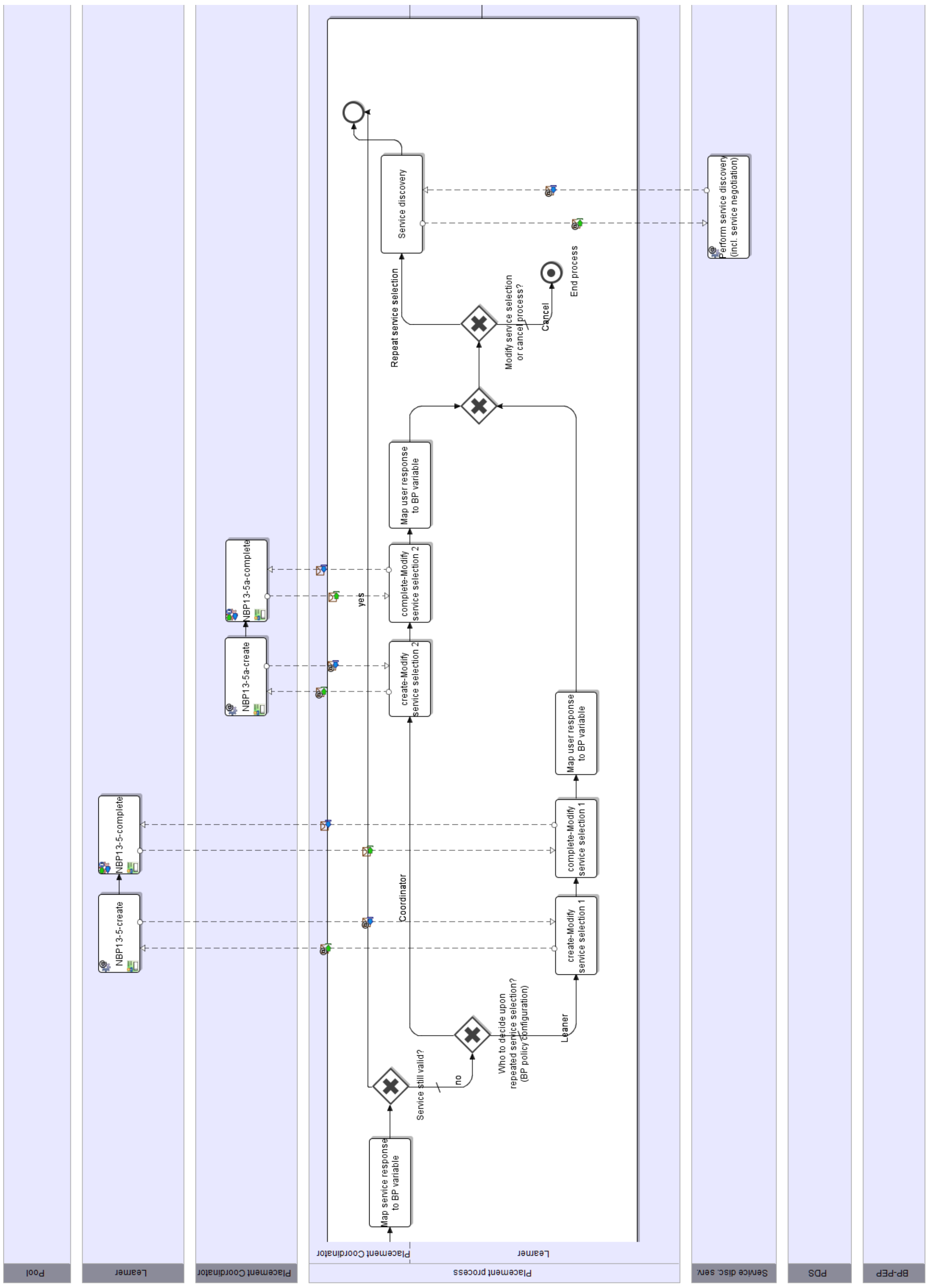

Pool
Learner
Placement Coordinator
Placement process
Placement Coordinator
Learner
Service disc. serv.
PDS
BP-PEP
Map service response to BP variable
Service still valid?
no
Who to decide upon repeated service selection? (BP policy configuration)
Coordinator
Leaner
create-Modify service selection 1
complete-Modify service selection 1
Map user response to BP variable
NBP13-5-create
NBP13-5-complete
create-Modify service selection 2
complete-Modify service selection 2
yes
NBP13-5a-create
NBP13-5a-complete
Map user response to BP variable
Modify service selection or cancel process?
Cancel
End process
Repeat service selection
Service discovery
Perform service discovery (incl. service negotiation)

Figure 3.17: Nottingham Student Placement Process Model – 7

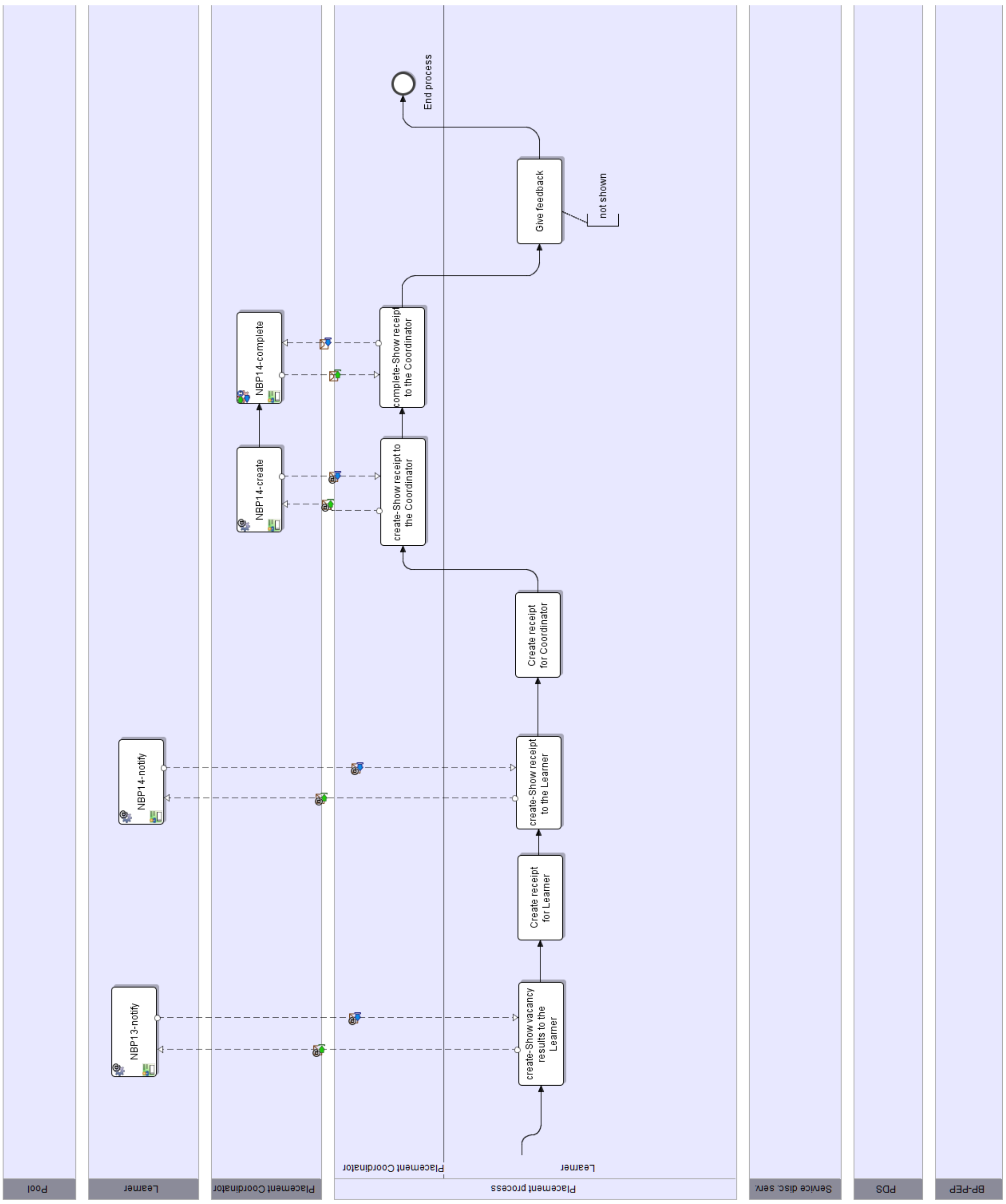

Pool
Learner
Placement Coordinator
Placement process
Placement Coordinator
Learner
Service disc. serv.
PDS
BP-PEP
NBP13-notify
NBP14-notify
NBP14-create
NBP14-complete
create-Show vacancy results to the Learner
Create receipt for Learner
create-Show receipt to the Learner
Create receipt for Coordinator
create-Show receipt to the Coordinator
complete-Show receipt to the Coordinator
Give feedback
not shown
End process

Figure 3.18: Nottingham Student Placement Process Model – 8

4 The Kenteq Use Case of Employability

4.1 Overview

The use case of this demonstrator is a Mass Layoff Scenario, where Kenteq acts as a service provider to coach employees in such a situation to plan their future employability, as one partner building the Tripod cooperation. More details are described in Deliverable 9.1 ([11]).

Tripod is a Dutch cooperation of employability partners, with the goal of better services and user centricity. They made an agreement on a national standard for exchange of ePortfolio data, NTA 2035[12], and agreed to use the same tables and vocabularies.

The following service providers are participating in Tripod and are integrated into the Mass Layoff business process:

- Kenteq: PCP/APL provider
- NewCarFactory: provides HR information
- UWV-werk.nl : Vacancy provider
- Paragin: ePortfolio provider

The objective of the Kenteq use case is to prove that the TAS^3 trust infrastructure can perform in a realistic scenario for which no current system exists. The demonstrator aims to demonstrate and improve the following areas:

- Usability of the general portal
- Single sign-on
- Policy management
- Data discovery
- Interaction of Multiple Service Providers
- Audit of multiple Service Providers
- Legacy data integration
- Improved Dashboard functionality

From the business process perspective the integration of multiple service providers is the main challenge, especially because these service providers are not only integrated using technical interfaces. In contrast to the Nottingham demonstrator the service providers Kenteq, UWV-werk.nl and Paragin provide user interfaces, as well. This means that the user has to be redirected between several websites, which is a challenge regarding user centricity.

Figure 4.1 gives an overview of the Mass Layoff business process, depicting the three main steps of the process.

In the first phase the HR (human resources) data of a user is being imported from his/her employer, in the demonstrator NewCarFactory, to Kenteq. The user also has the possibility to enter his/her data manually. Kenteq makes an assessment of the HR data.

The second phase contains importing the assessment data to the user's e-portfolio at Paragin. If he/she wants to check and edit his/her e-portfolio data, the user is redirected to Paragin.

As the last step the user has to accept the proposed vacancy provider UWV-werk.nl and is redirected to their website. Using the e-portfolio data the vacancy search can be performed.

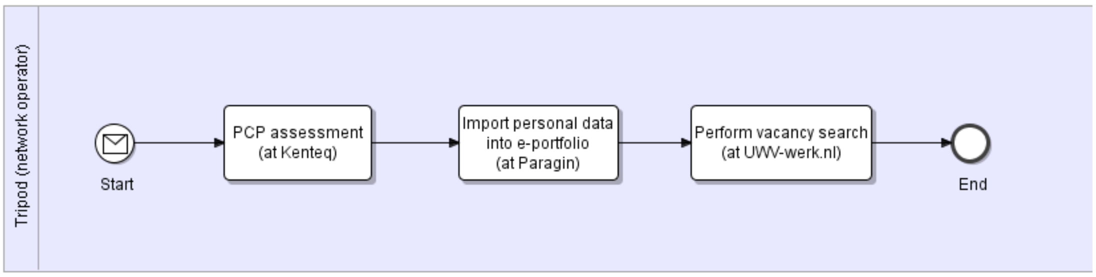

Tripod (network operator)
Start
PCP assessment (at Kenteq)
Import personal data into e-portfolio (at Paragin)
Perform vacancy search (at UWV-werk.nl)
End

Figure 4.1: Overview of the Kenteq Mass Layoff Process

4.2 Modelling the Employability Process

For developing the process model for the Kenteq business process we proceed in the same way as for the Nottingham demonstrators. We first model the process on the application level, then we complete the process model by adding security specifications as annotations, and finally we transform the security annotations into configuration parameters for business process components and into elements of the business process policy.

4.2.1 Kenteq Mass Layoff Process Model

The Kenteq application process model with security annotations is depicted in Figures 4.2 to 4.7. In this section we explain the process logic for the application, and we illustrate the security annotations in the next section.

Like the second Nottingham demonstrator process a web service call starts the Mass Layoff process. At the beginning, a welcome screen provides an overview of the succeeding steps to the user.

4.2.1.1 PCP assessment phase

The personal compentency profile (PCP) assessment phase starts with a discovery of the jobseekers HR data, which will be found at his last employer, NewCarFactory. The jobseeker is able to import this data to Kenteq, the Tripod provider for creating personal competency profiles. The jobseeker is then guided to the Kenteq website, which performs the assessment. After completing the assessment the jobseeker is guided back to the business process interface.

4.2.1.2 Import personal data into e-portfolio phase

In the import personal data into e-portfolio phase the jobseeker can transfer his/her assessment data to an e-portfolio provider, which is the Tripod partner Pargin. Again the jobseeker can search for existing data about him/her within the TAS^3 framework. Now, data will be found at NewCarFactory and Kenteq. The jobseeker wants to import his/her assessment data into the e-portfolio, so he/she permits the process to transfer the data. The jobseeker is guided to the website of Paragin where he/she can view and edit the data in the e-portfolio.

4.2.1.3 Perform vacancy search phase

The perform vacancy search phase consists of a user information and a forwarding task in the business process. The vacancy search itself is performed at UWV-werk.nl, where the jobseeker is forwarded to.

4.2.2 Security Annotated Process Model

In the following, we describe the security annotations of the Kenteq Use Case of Employability (from process start (left side in Figure 4.2) to process end (right side in Figure 4.7)):

Lane *Jobseeker* (Figure 4.3):

The annotation term
≪Authn attribute="(company,NewCarFactory),(position,employee)" idp="idp.tas3.eu"≫
implies that all process participants of role "Jobseeker" have to be authenticated at the IdP *idp.tas3.eu* as employees of NewCarFactory.

≪BoD≫: Within one process instance all activities of lane "Jobseeker" have to be performed the same user. A re-assignment is not allowed.

≪Assignment: type="Role"≫ means that the lane name "Jobseeker" represents a role with the name "Jobseeker" and that all activities of lane "Jobseeker" have to be performed by role holders of "Jobseeker".

Start event:

The security annotation *≪Audit policy="auditAll"≫* specifies that the audit policy *auditAll* has to be applied. This means that all security-related events will be logged.

Activity *complete-Welcome screen with explanation for the steps of the BP*:

There are three annotations enforcing user involvements. The first annotation,
≪UInvolve:
type="Consent"
target="TAS3 PrivacyTerms"
display="yes,no"≫
controls that a user has to give consent to the TAS^3 Privacy Terms. He/she can select "yes" or "no".

The second user involvement,
≪UInvolve:
type="Consent"
target="Kenteq PCP T&C"
display="yes,no"≫
asks a user to give consent to the "Kenteq PCP T&C".

The third user request,
≪UInvolve:
type="Consent"
target="Kenteq APL/PCP "
display="yes,no"≫
asks interactively a user to give consent to the "Kenteq APL/PCP".

Activity *Search for jobseeker's HR data* (Figure 4.3):

The annotation
≪Conf: spec="encrypted,signed"≫
specifies that the message flow between the activity "Search for jobseeker's HR data" and the activity "Discover jobseeker's HR data" will be protected by means of encryption. The messages will be digitally signed to assure the identity of the originators. In the Kenteq Employability Process all message flows to external services must be encrypted and signed.

Activity *Export HR data to Kenteq*:

With the annotation
≪D-Delegation:
rights="write"
object="HR data of Jobseeker"
receiver="Kenteq"
spec="grant"≫

a delegation of ”write” access rights is specified from the jobseeker to the *Kenteq* webservice. This enables the Kenteq webservice to store the jobseeker’s HR data in the name of the jobseeker. Propagation of the delegated access rights is allowed, because subsystems or third party systems may me involved in storing the jobseekers data.

In Figure 4.5 there are two user involvements for tasks of lane ”Jobseeker”, aiming to give consent to ”Paragin e-portfolio Terms and Conditions” and to ”Paragin e-portfolio policy”.

There is another annotation for the delegation of data access rights, as described for Figure 4.3.

Further, two annotated user involvements are inserted in Figure 4.7 to let the user give consent to ”UWV-werk.nl” terms and conditions and the corresponding policy.

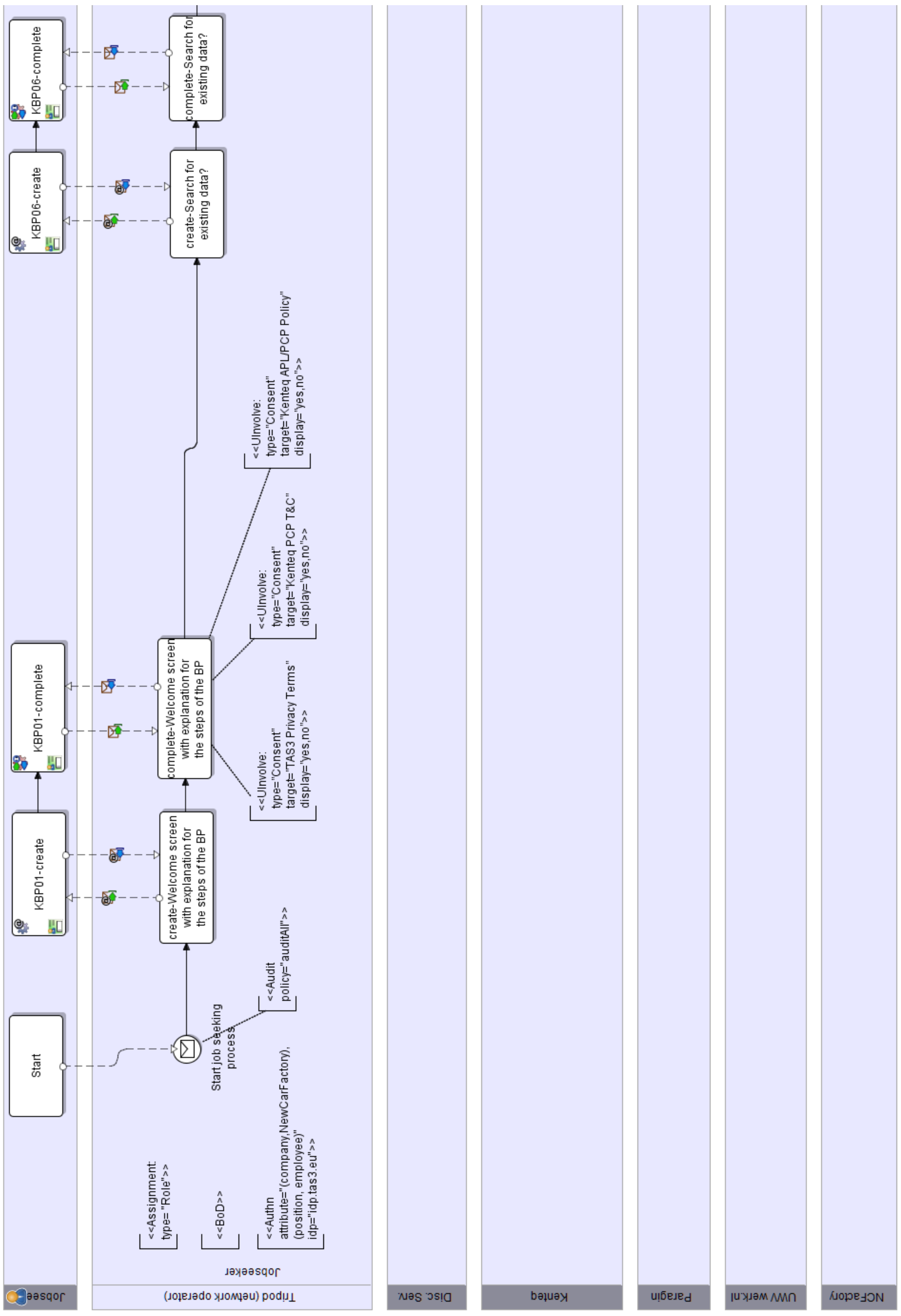


Figure 4.2: Kenteq Employability Process with Security Annotations – 1

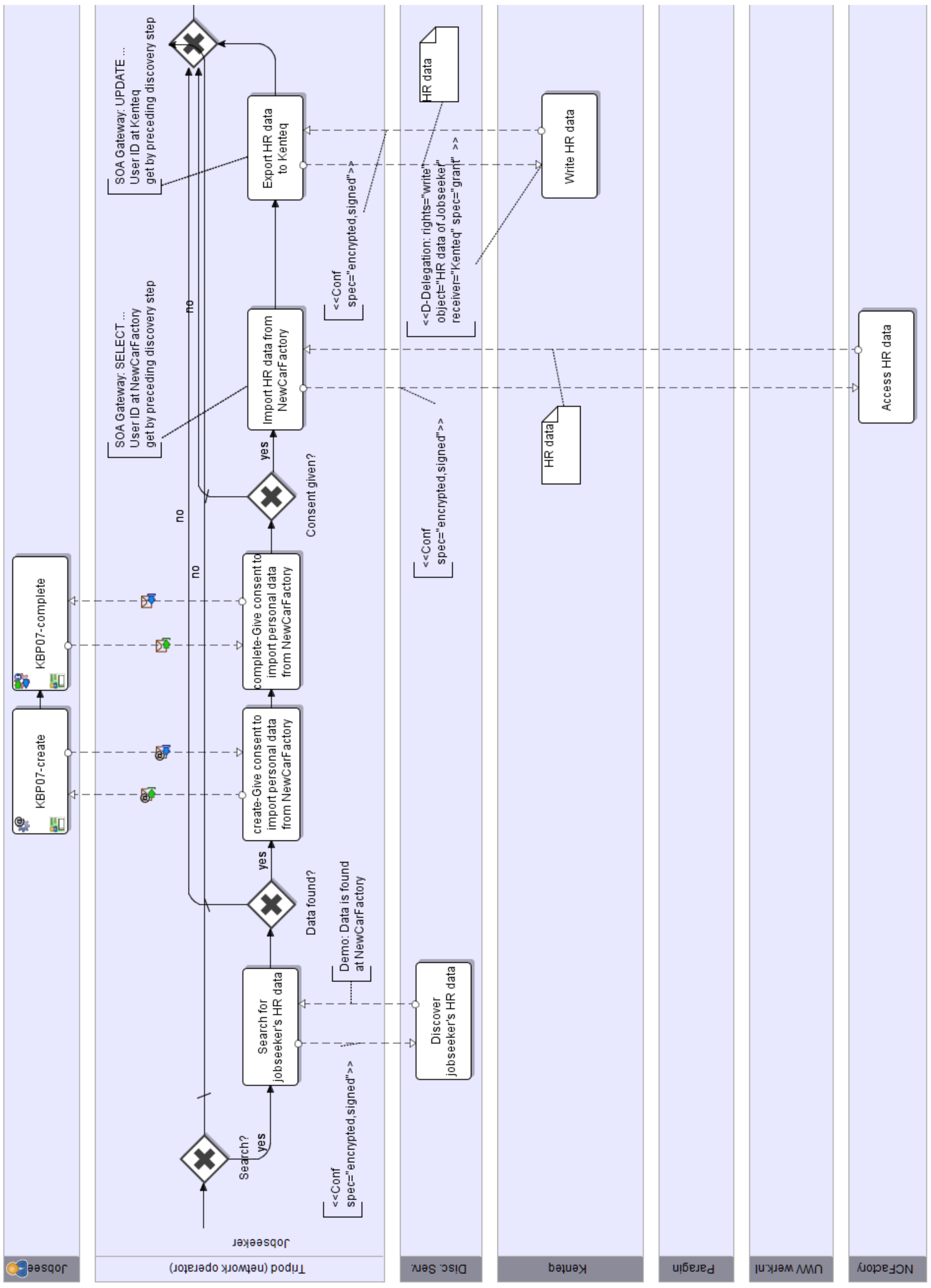

Jobseeker
Tripod (network operator)
Disc. Serv.
Kenteq
Paragin
UWV werk.nl
NCFactory
Jobseeker
Search?
yes
Search for jobseeker's HR data
<<Conf spec="encrypted,signed">>
Discover jobseeker's HR data
Demo: Data is found at NewCarFactory
Data found?
yes
no
create-Give consent to import personal data from NewCarFactory
KBP07-create
complete-Give consent to import personal data from NewCarFactory
KBP07-complete
Consent given?
yes
no
Import HR data from NewCarFactory
SOA Gateway: SELECT ... User ID at NewCarFactory get by preceding discovery step
<<Conf spec="encrypted,signed">>
HR data
Access HR data
Export HR data to Kenteq
SOA Gateway: UPDATE ... User ID at Kenteq get by preceding discovery step
<<Conf spec="encrypted,signed">>
<<D-Delegation: rights="write" object="HR data of Jobseeker" receiver="Kenteq" spec="grant" >>
HR data
Write HR data

Figure 4.3: Kenteq Employability Process with Security Annotations – 2

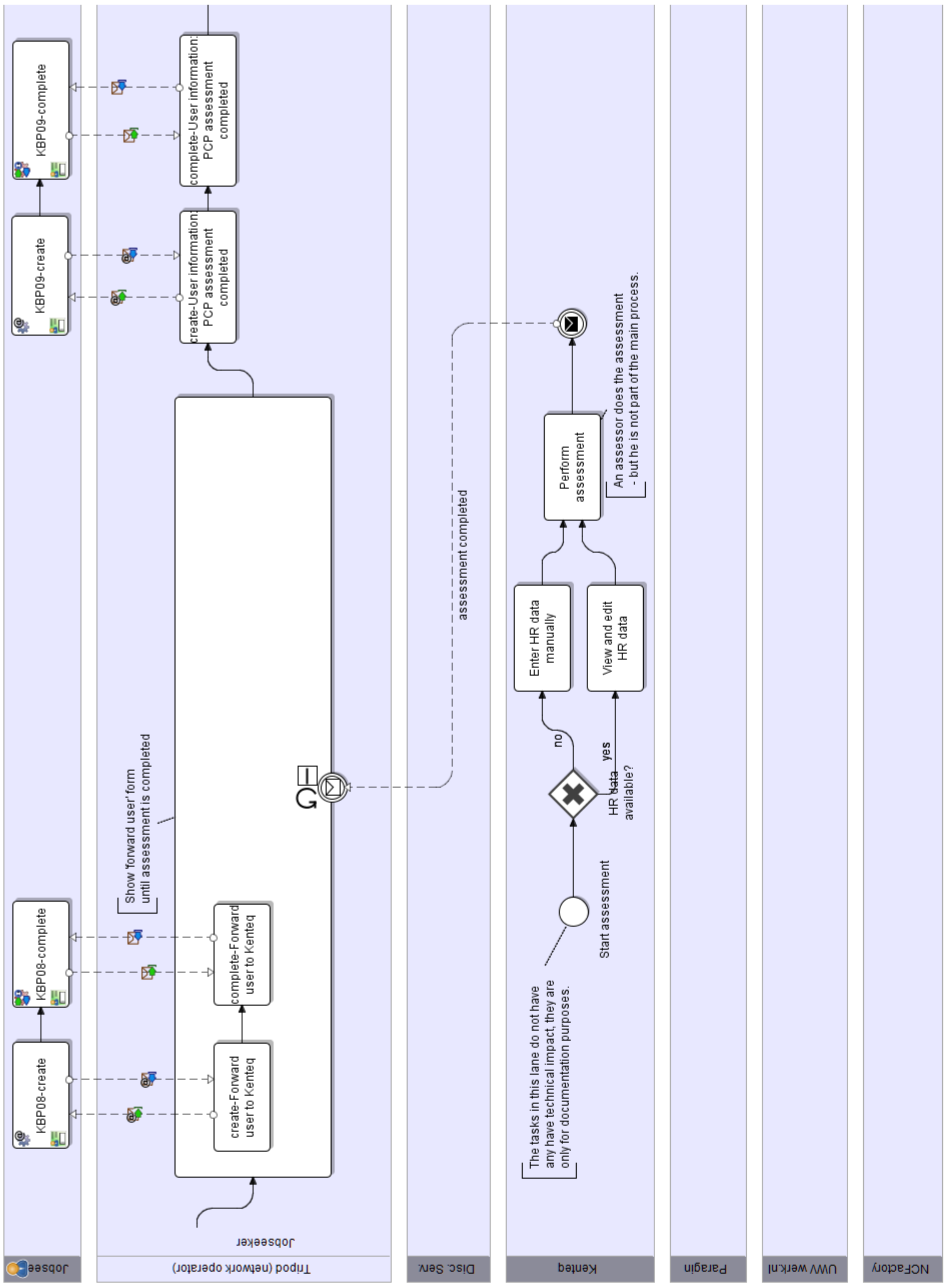

Jobseeker
Tripod (network operator)
Disc. Serv.
Kenteq
Paragin
UWV werk.nl
NOFactory
Jobseeker
KBP08-create
KBP08-complete
KBP09-create
KBP09-complete
create-Forward user to Kenteq
complete-Forward user to Kenteq
Show 'forward user' form until assessment is completed
create-User information: PCP assessment completed
complete-User information: PCP assessment completed
assessment completed
The tasks in this lane do not have any have technical impact, they are only for documentation purposes.
Start assessment
HR data available?
no
yes
Enter HR data manually
View and edit HR data
Perform assessment
An assessor does the assessment - but he is not part of the main process.

Figure 4.4: Kenteq Employability Process with Security Annotations – 3

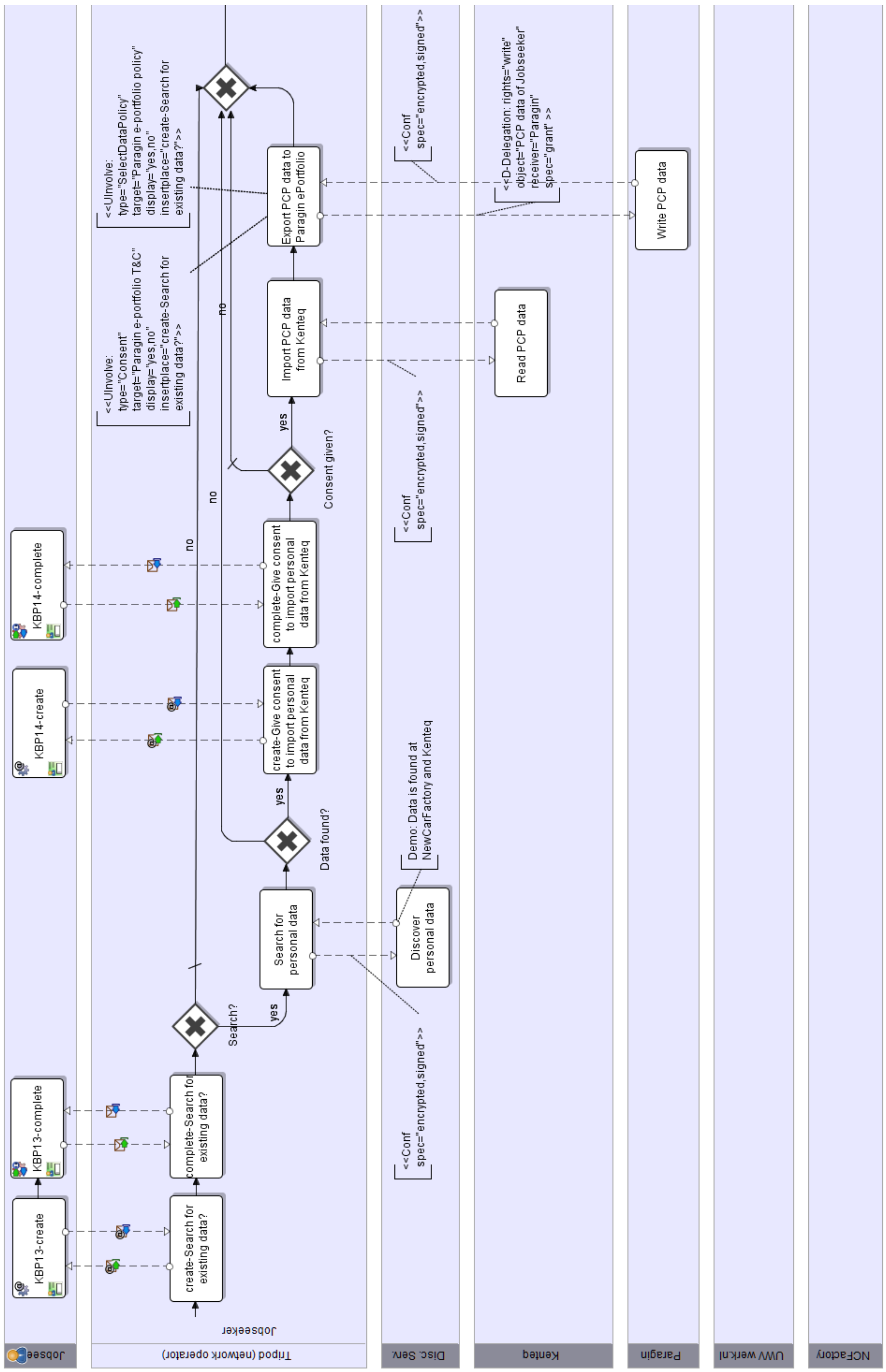

Jobseeker
Tripod (network operator)
Jobseeker
Disc. Serv.
Kenteq
Paragin
UWV werk.nl
NCFactory
KBP13-create
KBP13-complete
KBP14-create
KBP14-complete
create-Search for existing data?
complete-Search for existing data?
Search?
yes
no
Search for personal data
Data found?
yes
no
create-Give consent to import personal data from Kenteq
complete-Give consent to import personal data from Kenteq
Consent given?
yes
no
Import PCP data from Kenteq
Export PCP data to Paragin ePortfolio
Discover personal data
Read PCP data
Write PCP data
<<Conf spec="encrypted,signed">>
Demo: Data is found at NewCarFactory and Kenteq
<<Conf spec="encrypted,signed">>
<<Conf spec="encrypted,signed">>
<<D-Delegation: rights="write" object="PCP data of Jobseeker" receiver="Paragin" spec="grant" >>
<<UIinvolve: type="Consent" target="Paragin e-portfolio T&C" display="yes,no" insertplace="create-Search for existing data?">>
<<UIinvolve: type="SelectDataPolicy" target="Paragin e-portfolio policy" display="yes,no" insertplace="create-Search for existing data?">>

Figure 4.5: Kenteq Employability Process with Security Annotations – 4

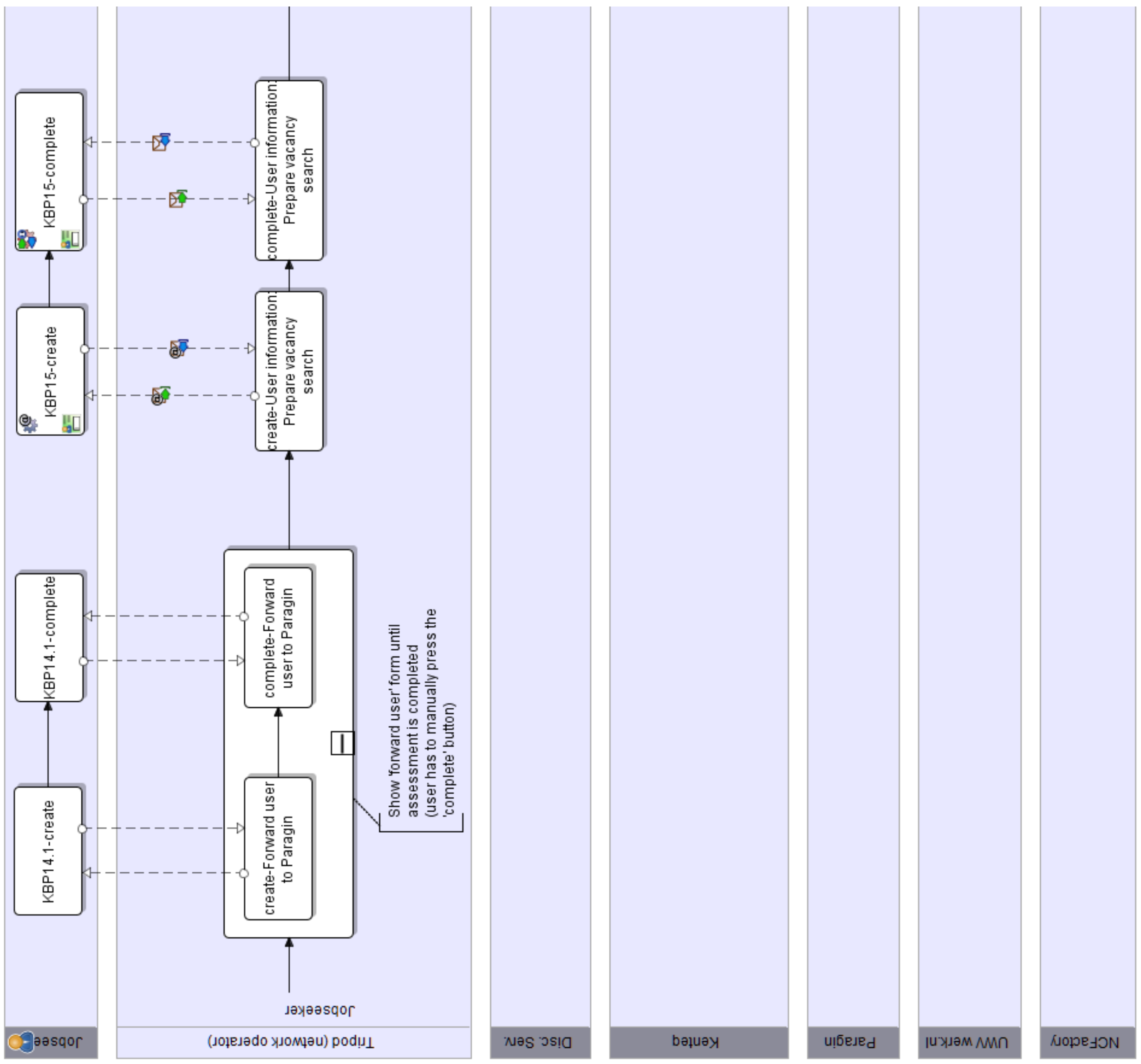

Jobseeker
Tripod (network operator)
Disc. Serv.
Kenteq
Paragin
UWV werk.nl
NOFactory
KBP14.1-create
KBP14.1-complete
KBP15-create
KBP15-complete
create-Forward user to Paragin
complete-Forward user to Paragin
create-User information: Prepare vacancy search
complete-User information: Prepare vacancy search
Jobseeker
Show 'forward user' form until assessment is completed (user has to manually press the 'complete' button)

Figure 4.6: Kenteq Employability Process with Security Annotations – 5

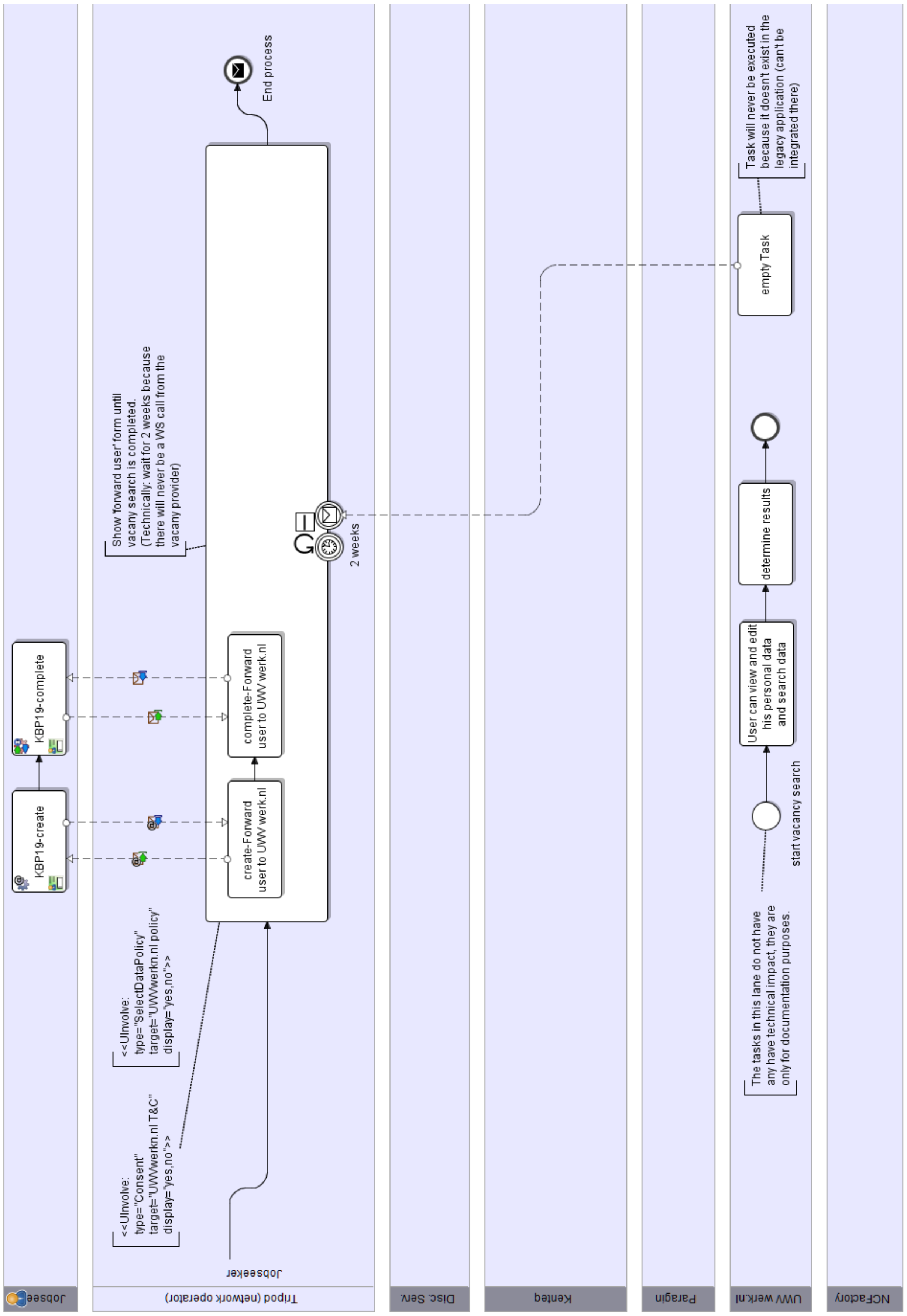

Jobseeker
Tripod (network operator)
Disc. Serv.
Kenteq
Paragin
UWV werk.nl
NCFactory
Jobseeker
<<UInvolve: type="Consent" target="UWVwerkn.nl T&C" display="yes,no">>
<<UInvolve: type="SelectDataPolicy" target="UWVwerkn.nl policy" display="yes,no">>
KBP19-create
KBP19-complete
create-Forward user to UWV werk.nl
complete-Forward user to UWV werk.nl
Show 'forward user' form until vacany search is completed. (Technically: wait for 2 weeks because there will never be a WS call from the vacany provider)
2 weeks
End process
The tasks in this lane do not have any have technical impact, they are only for documentation purposes.
start vacancy search
User can view and edit his personal data and search data
determine results
empty Task
Task will never be executed because it doesn't exist in the legacy application (can't be integrated there)

Figure 4.7: Kenteq Employability Process with Security Annotations – 6

4.3 Operational sequence and involved components

All software components that are being developed by WP3, except the T3-BP-MGR, will be used in the Kenteq demonstrator. These are:

- Business Process Security Modelling and Configuration Component (T3-BP-SMC)
- Apache ODE Business Process Execution Engine (T3-BP-ENGINE-ODE)
- Business Process User Client (T3-BP-CLIENT)
- Business Process Policy Enforcement Point (T3-PEP-BP)
- Business Process Policy Decision Point (T3-PDP-BP)
- Business Process Policy Information Point (T3-BP-PIP)
- Business Process Delegation Service (T3-BP-DEL)
- Business Process Administration Component (T3-BP-MGR)

The components are described in detail in Deliverable D3.2 [4].

The following components and services developed by TAS^3 project partners will also be integrated into the business process of Kenteq demonstrator:

- T3-DEL: Delegation Service, developed by KENT T3-LOG-SAWS: Secure Auditing Web Service, developed by KENT
- T3-PDP-M: Policy Decision Point/Master PDP, developed by KENT
- T3-PDP-P: PERMIS PDP, developed by KENT
- T3-PDP-T: Trust Policy Decision Point, developed by TUE
- T3-SP-MATCHER: Job Profile Matching Service, developed by NOT
- T3-ZXID-LINUX-X86: ZXID library and tools for TAS3 in Apache, Java, PHP, and C, developed by RIS
- T3-ZXID-SRC: Sources for the ZXID package, developed by RIS

4.4 Implementation

In this section we describe the current status of the implementation of the Kenteq demonstrator. At the current stage of integration not all issues are clarified yet. I.e., the final version of the demonstrator will have some additional or changed parts. We will provide a description and validation in the next project period.

4.4.1 PCP assessment phase

The Tripod website redirects the user (jobseeker) to the dashboard containing the business process user interface. A web service call starts an instance of the Kenteq Mass Layoff process, that is triggered by a jobseeker interaction with the Tripod website (Fig. 4.8). The user logs in at Tripod using single sign-on. After logging in he/she is forwarded to the business process interface at the dashboard, where he/she is automatically logged in.

Single sign-on is executed the same way as in both Nottingham demonstrators, using ZXID (T3-ZXID-SRC), an identity provider and a modified version of Intalio Tempo (T3-BP-CLIENT) on the business process management system side.

The jobseeker lane in the business process has been annotated with a *BoD (Binding of Duty)* constraint, which means that for one process instance all tasks in this lane have to be performed by the same user. This means the user logging in and starting the process instance will execute all other tasks of the Kenteq process. Assignment of users is implemented in the T3-BP-CLIENT component, which controls everything related to subjects interacting with the business process. Assignments of users to tasks need to be validated by the Business Process PDP. The T3-PDP-BP accesses the Business Process PIP (T3-BP-PIP), which stores already existing assignments of users to tasks, and the business process policy, which contains process-specific security constraints like binding of duty.

The first tasks in the process ask the user to agree to TAS^3 and to Kenteq terms&conditions as well as to the Kenteq APL/PCP policy. At the final stage of development this tasks should automatically be generated by transforming the security annotations *<<RequestToUser: type="Consent" target="TAS3 Privacy Terms" display="yes,no">>*, *<<RequestToUser: type="Consent" target="Kenteq PCP T&C" display="yes,no">>* and *<<RequestToUser: type="Consent" target="Kenteq APL/PCP Policy" display="yes,no">>* into business process elements. This should be accomplished by the T3-BP-SMC component. At the current stage of implementation transformation is implemented only for some security annotations.

If the user agrees to search for personal data about him/her within the TAS^3 network, the search is performed by the ZXID component (T3-ZXID-SRC) (see Figure 4.10 - 4.11). The user is able to import the discovered data to Kenteq. Therefor an interface for data import at Kenteq as well as an interface for data export at the service provider where data has been discovered is required.

For perfoming the assessment the user is forwarded to the Kenteq website, where he is guided through the assessment steps (Fig. 4.12). The assessment steps themselves are not under the control of the business process. While integration of legacy systems into service-oriented architectures is normally a technical problem, in the Kenteq demonstrator it is a challenge regarding the user interface as well. The user interfaces of the Intalio business process management system and the user interfaces of the Kenteq website as well as the websites of the other involved partners have to be integrated to let the user have a feeling of moving through one single jobseeker process, not through several different subsystems coupled together. Apart from the design of the user screens there is the problem of forwarding the user between different service provider websites including the business process management system. Regarding Kenteq this is a minor problem: A forwarding business process screen with a link to the Kenteq website is shown to the user. This screen remains open until a web service call from Kenteq notifies the BPMS about the assessment being completed. This guarantees that the user always finds his/her way back to Kenteq during the assessment, even in case he/she closes the browser window with the Kenteq website opened.

4.4.2 Import personal data into e-portfolio phase

From a technical point of view this phase is quite similar to last part of the PCP assessment phase (see Figures 4.13 to 4.16). Personal data of the user is imported to Pargin, the e-portfolio provider in this scenario. Then the user is forwarded to Paragin to review his data. Regarding the forwarding between the systems we chose an approach similiar to the one used for the Kenteq forwarding. But different to the Kenteq appropach, for the demo it is not possible to integrate a web service call into the Paragin system, which indicates the end of the user's data review and triggers the business process to continue. For this reason the user has to manually tell the process via a user screen interaction that the review at Pargin is completed.

4.4.3 Perform vacancy search phase

Except the user interactions regarding terms & conditions and policy agreement (Fig. 4.16 and 4.17) this phase is mainly being performed at the website of UWV-werk.nl, the vacancy provider. The user is forwarded to UWV-werk.nl, where the jobseeker process is being completed from the user point of view (Fig. 4.18). As with the Paragin system it is not possible to integrate a web service call into the UWV-werk.nl website, just for the purpose of the demo. Because of this the business process on the BPMS is technically terminated two weeks after the user has been forwarded to UWV-werk.nl. This is no solution suitable for a real world operation of the process, but due to limited software engineering ressources a full integration solution cannot be realised at the moment.

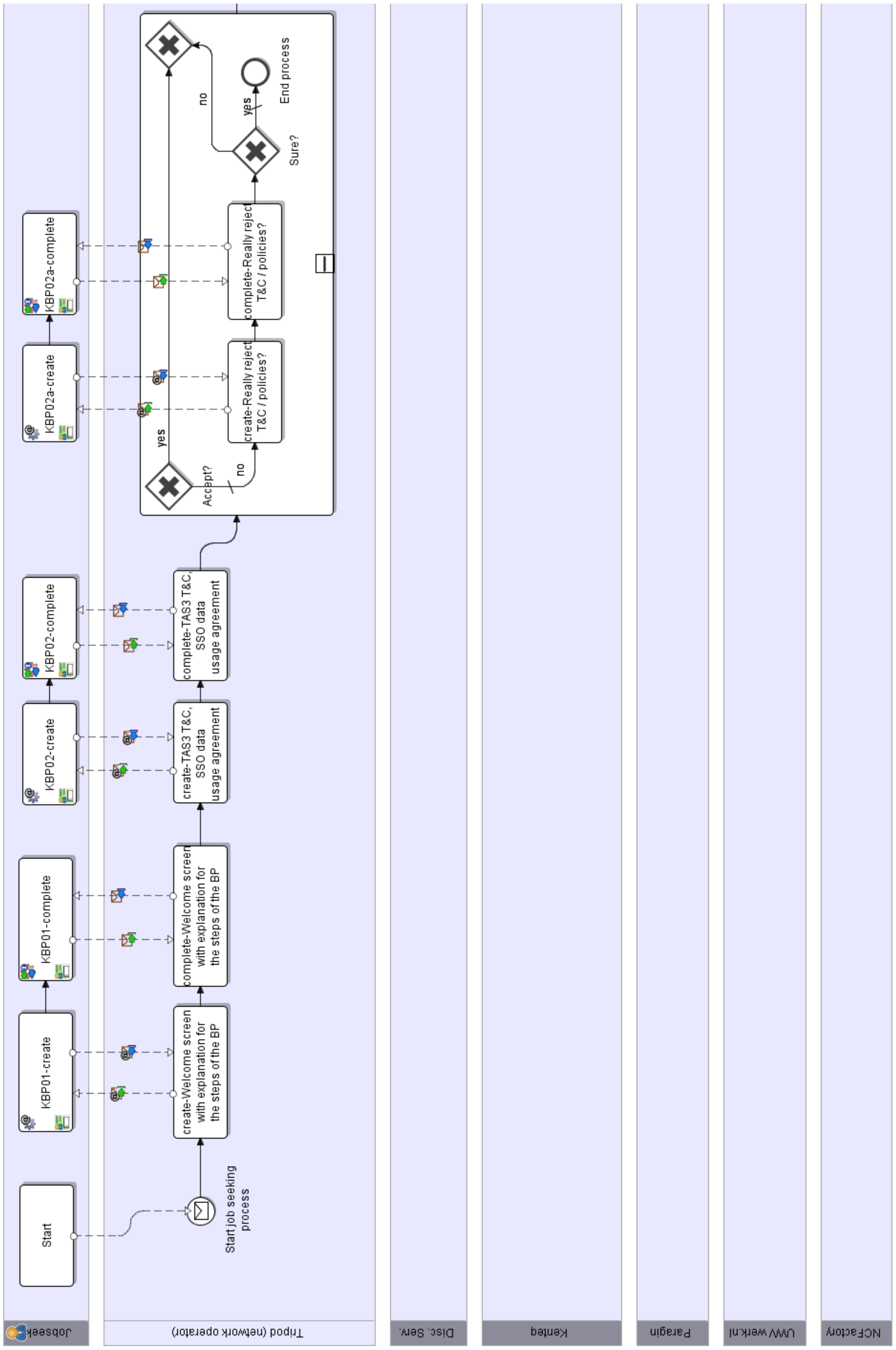

Jobseek
Tripod (network operator)
Disc. Serv.
Kenteq
Paragin
UWV werk.nl
NCFactory
Start
Start job seeking process
KBP01-create
KBP01-complete
create-Welcome screen with explanation for the steps of the BP
complete-Welcome screen with explanation for the steps of the BP
KBP02-create
KBP02-complete
create-TAS3 T&C, SSO data usage agreement
complete-TAS3 T&C, SSO data usage agreement
Accept?
yes
no
KBP02a-create
KBP02a-complete
create-Really reject T&C / policies?
complete-Really reject T&C / policies?
Sure?
yes
no
End process

Figure 4.8: Kenteq Employability Process Model – 1

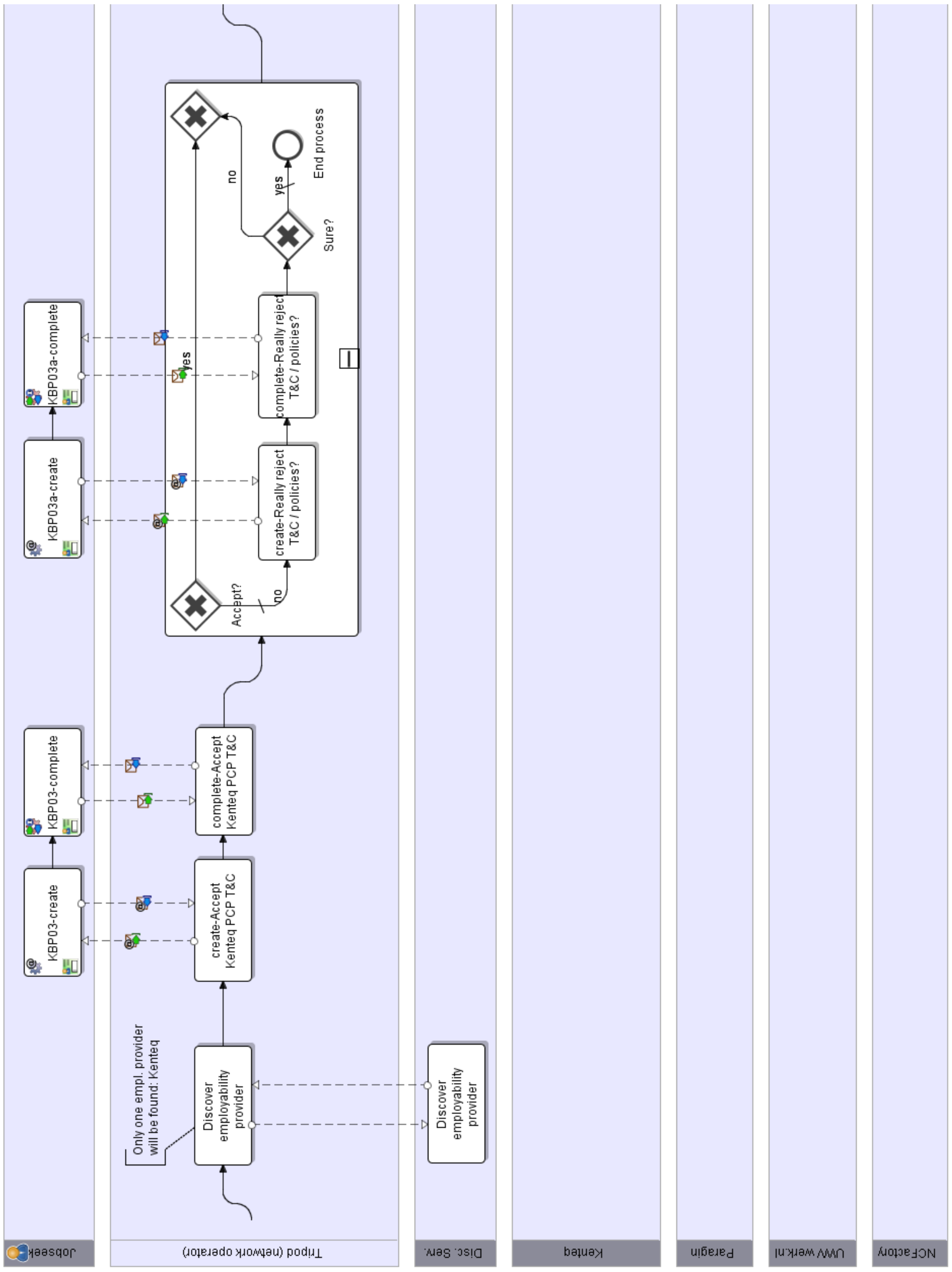

Jobseek
Tripod (network operator)
Disc. Serv.
Kenteq
Paragin
UWV werk.nl
NCFactory
Discover employability provider
Only one empl. provider will be found: Kenteq
Discover employability provider
create-Accept Kenteq PCP T&C
complete-Accept Kenteq PCP T&C
KBP03-create
KBP03-complete
Accept?
no
create-Really reject T&C / policies?
complete-Really reject T&C / policies?
yes
KBP03a-create
KBP03a-complete
Sure?
yes
no
End process

Figure 4.9: Kenteq Employability Process Model – 2

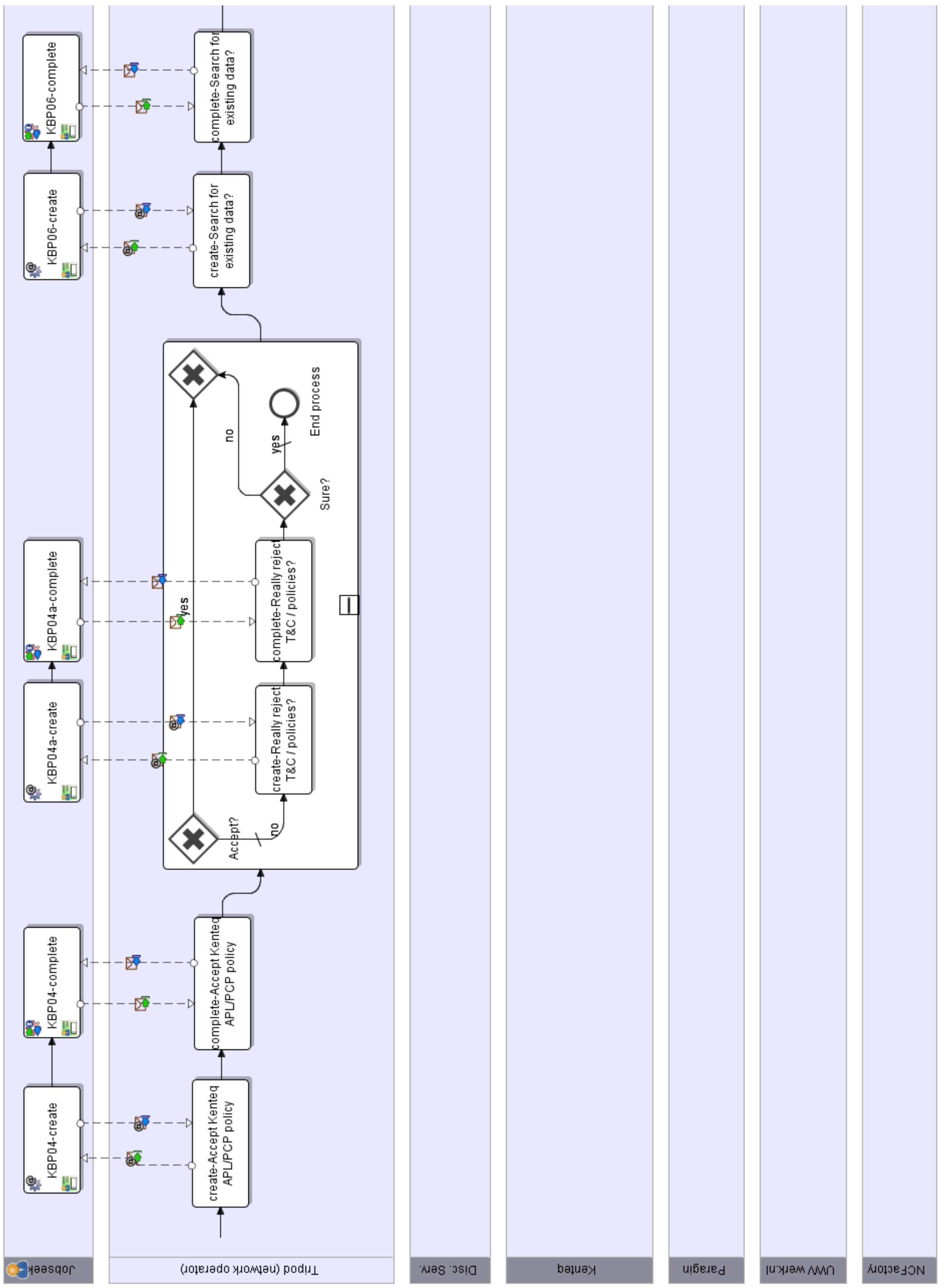

Jobseek
Tripod (network operator)
Disc. Serv.
Kenteq
Paragin
UWV werk.nl
NCFactory
KBP04-create
KBP04-complete
KBP04a-create
KBP04a-complete
KBP06-create
KBP06-complete
create-Accept Kenteq APL/PCP policy
complete-Accept Kenteq APL/PCP policy
Accept?
no
yes
create-Really reject T&C / policies?
complete-Really reject T&C / policies?
Sure?
yes
no
End process
create-Search for existing data?
complete-Search for existing data?

Figure 4.10: Kenteq Employability Process Model – 3

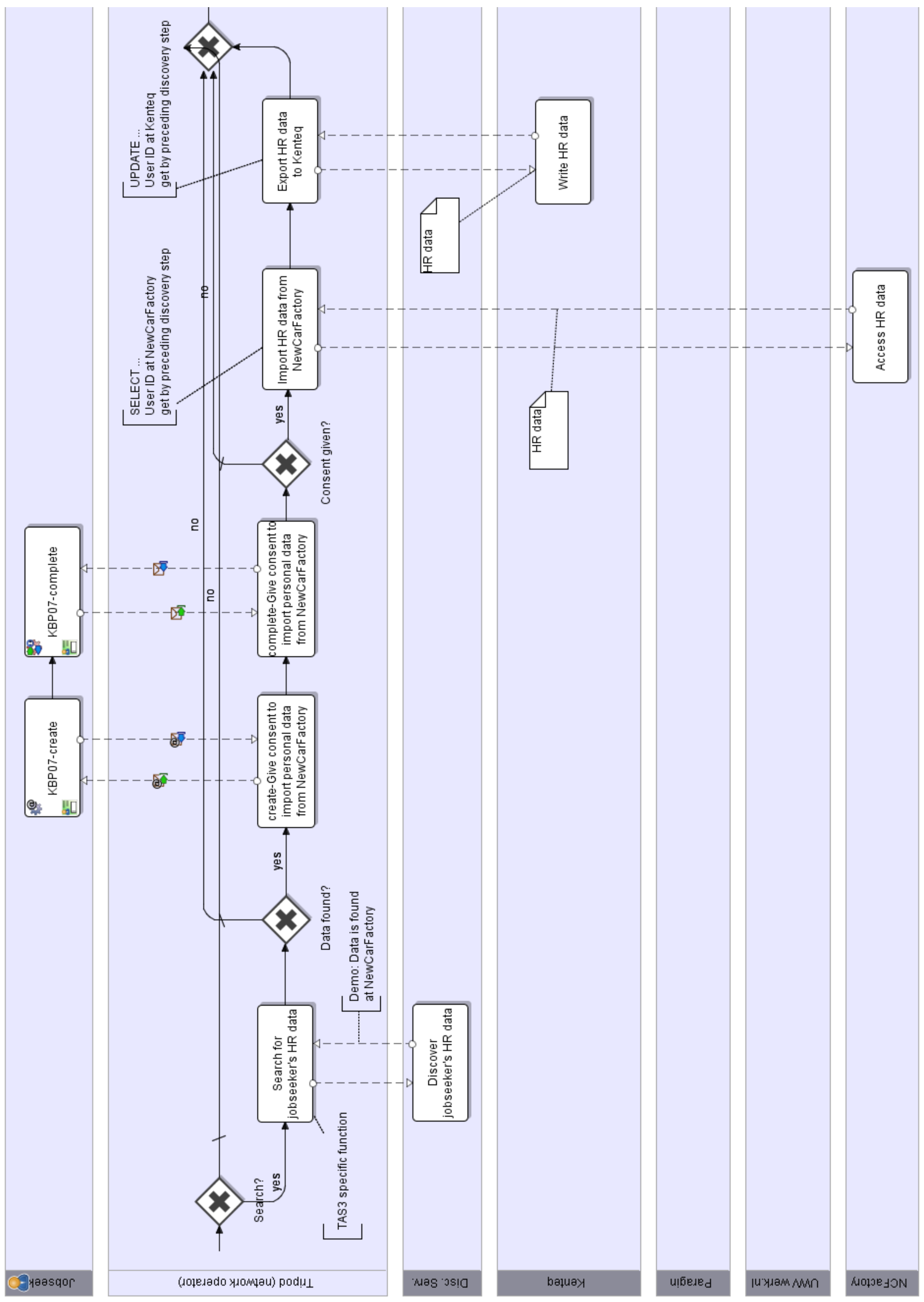

Jobseeker
Tripod (network operator)
Disc. Serv.
Kenteq
Paragin
UWV werk.nl
NCFactory
Search?
yes
TAS3 specific function
Search for jobseeker's HR data
Discover jobseeker's HR data
Demo: Data is found at NewCarFactory
Data found?
yes
no
create-Give consent to import personal data from NewCarFactory
KBP07-create
complete-Give consent to import personal data from NewCarFactory
KBP07-complete
no
no
Consent given?
yes
Import HR data from NewCarFactory
SELECT ... User ID at NewCarFactory get by preceding discovery step
Access HR data
HR data
no
Export HR data to Kenteq
UPDATE ... User ID at Kenteq get by preceding discovery step
Write HR data
HR data

Figure 4.11: Kenteq Employability Process Model – 4

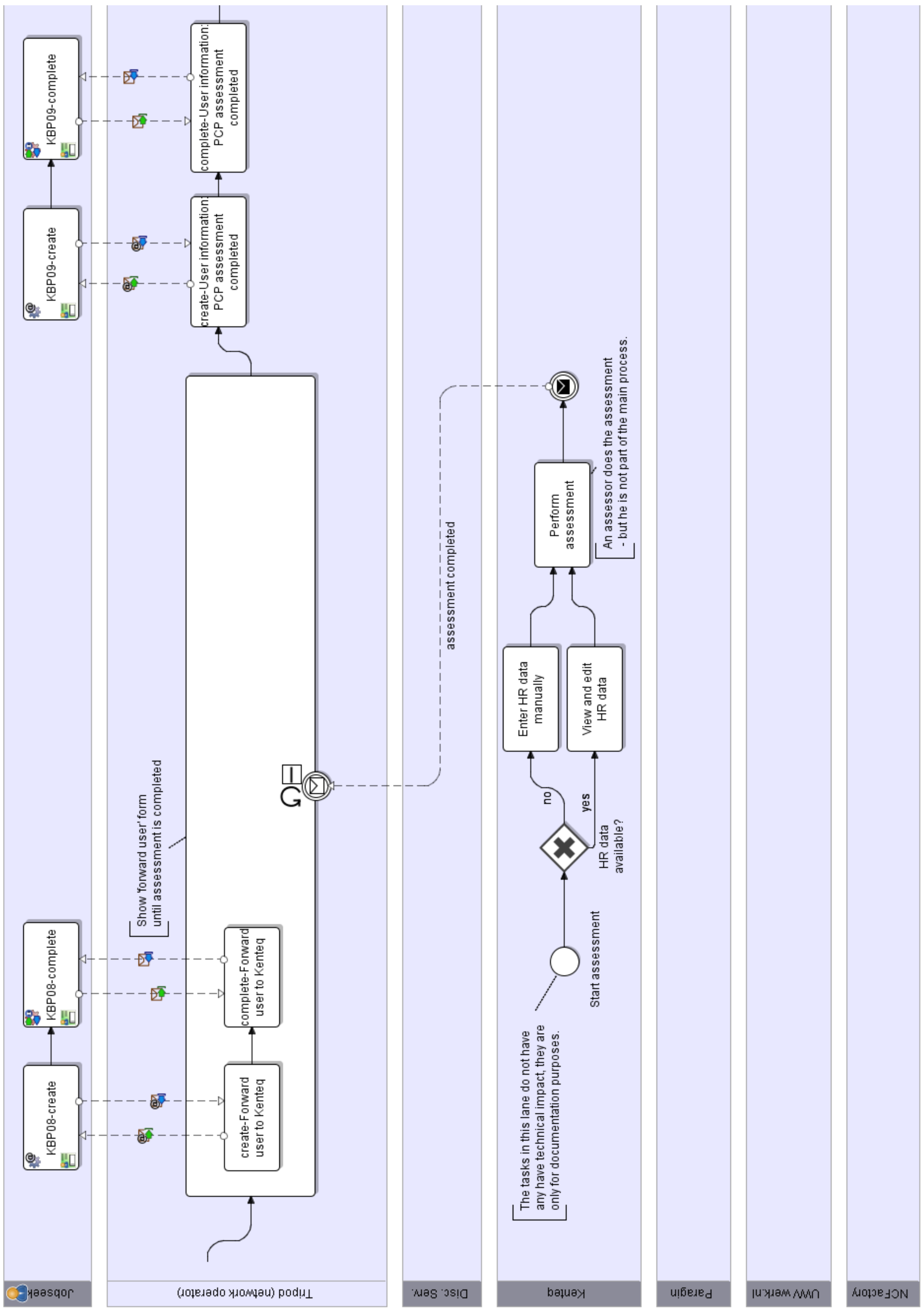

JobSeek
Tripod (network operator)
Disc. Serv.
Kenteq
Paragin
UWV werk.nl
NCFactory
KBP08-create
KBP08-complete
KBP09-create
KBP09-complete
create-Forward user to Kenteq
complete-Forward user to Kenteq
Show 'forward user' form until assessment is completed
create-User information: PCP assessment completed
complete-User information: PCP assessment completed
assessment completed
The tasks in this lane do not have any have technical impact, they are only for documentation purposes.
Start assessment
HR data available?
no
yes
Enter HR data manually
View and edit HR data
Perform assessment
An assessor does the assessment - but he is not part of the main process.

Figure 4.12: Kenteq Employability Process Model – 5

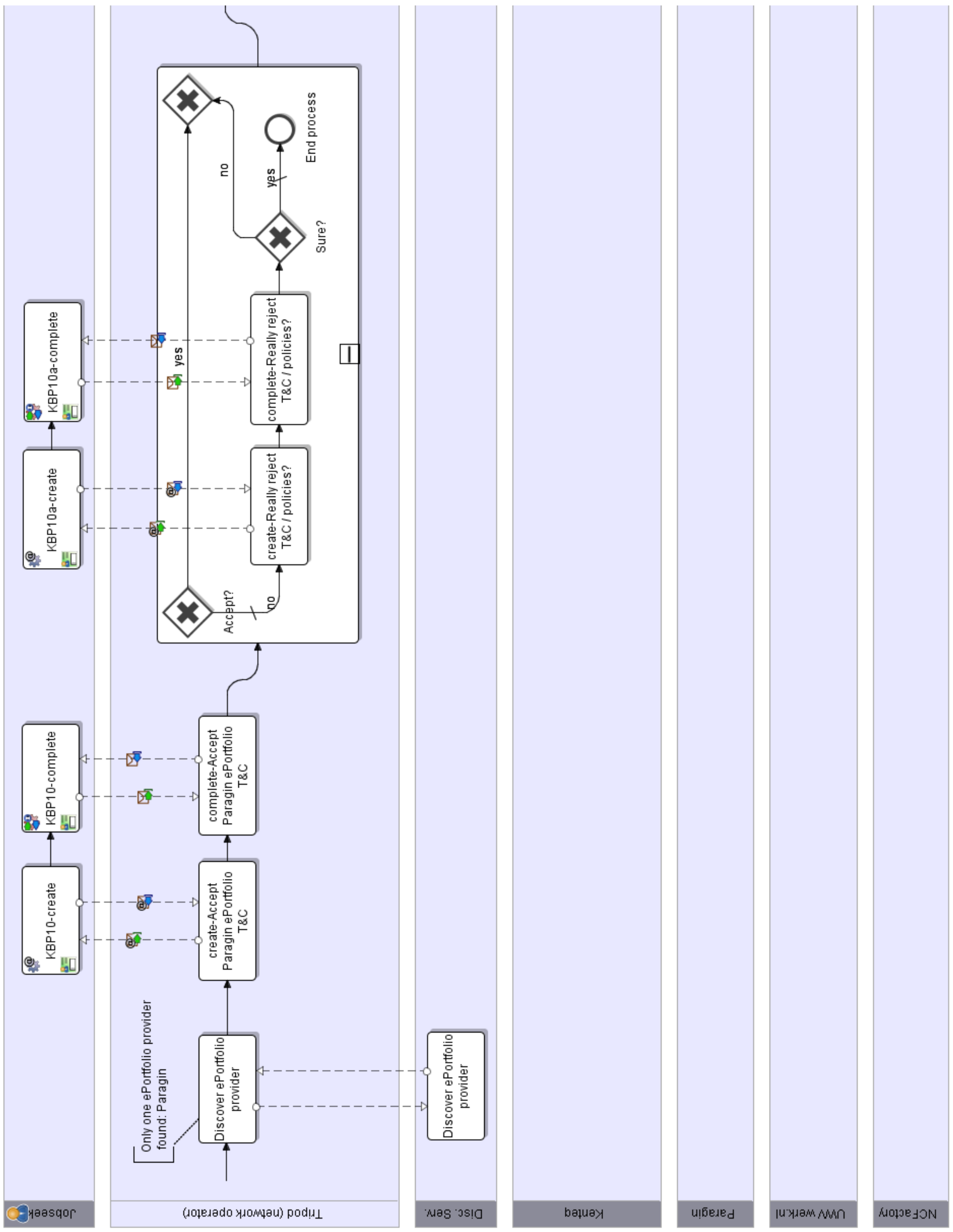

Jobseek
Tripod (network operator)
Disc. Serv.
Kenteq
Paragin
UWV werk.nl
NCFactory
KBP10-create
KBP10-complete
KBP10a-create
KBP10a-complete
Only one ePortfolio provider found: Paragin
Discover ePortfolio provider
create-Accept Paragin ePortfolio T&C
complete-Accept Paragin ePortfolio T&C
Accept?
no
yes
create-Really reject T&C / policies?
complete-Really reject T&C / policies?
Sure?
yes
no
End process
Discover ePortfolio provider

Figure 4.13: Kenteq Employability Process Model – 6

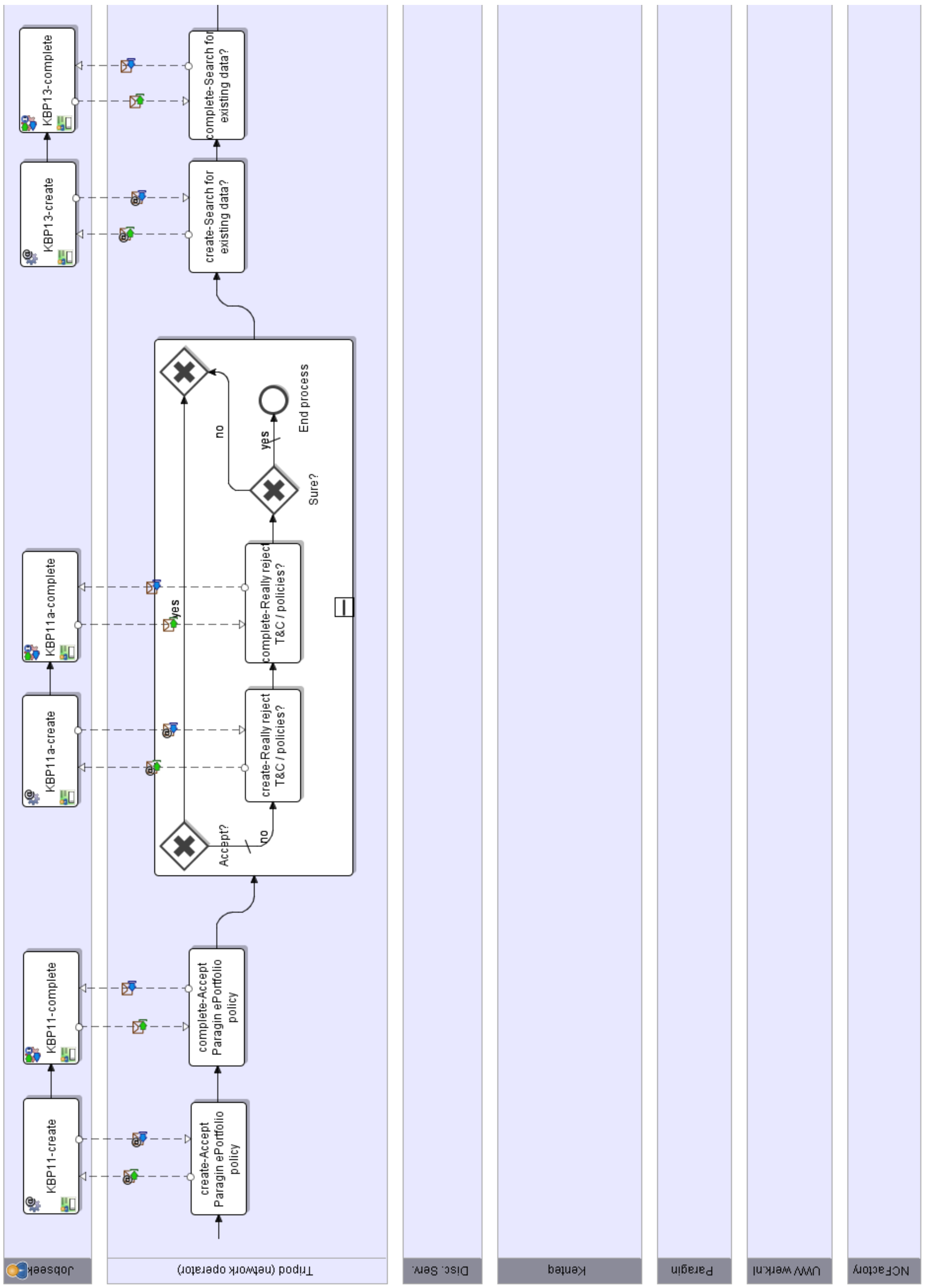

Jobseek
Tripod (network operator)
Disc. Serv.
Kenteq
Paragin
UWV werk.nl
NCFactory
KBP11-create
KBP11-complete
create-Accept Paragin ePortfolio policy
complete-Accept Paragin ePortfolio policy
Accept?
no
yes
KBP11a-create
KBP11a-complete
create-Really reject T&C / policies?
complete-Really reject T&C / policies?
Sure?
yes
End process
no
KBP13-create
KBP13-complete
create-Search for existing data?
complete-Search for existing data?

Figure 4.14: Kenteq Employability Process Model – 7

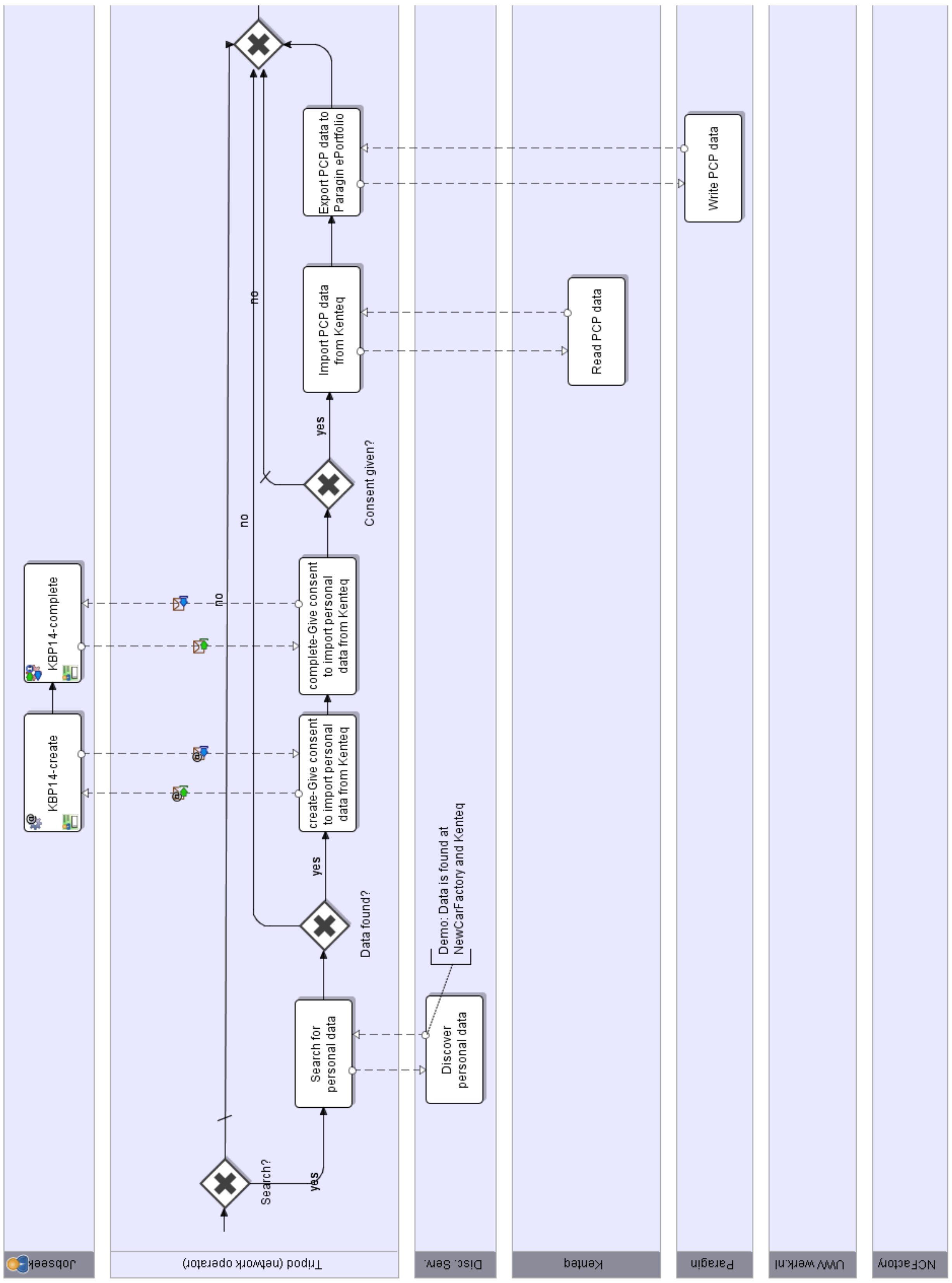

Jobseeker
Tripod (network operator)
Disc. Serv.
Kenteq
Paragin
UWV werk.nl
NCFactory
Search?
yes
Search for personal data
Discover personal data
Demo: Data is found at NewCarFactory and Kenteq
Data found?
yes
no
create-Give consent to import personal data from Kenteq
KBP14-create
complete-Give consent to import personal data from Kenteq
KBP14-complete
no
Consent given?
yes
no
Import PCP data from Kenteq
Read PCP data
Export PCP data to Paragin ePortfolio
Write PCP data

Figure 4.15: Kenteq Employability Process Model – 8

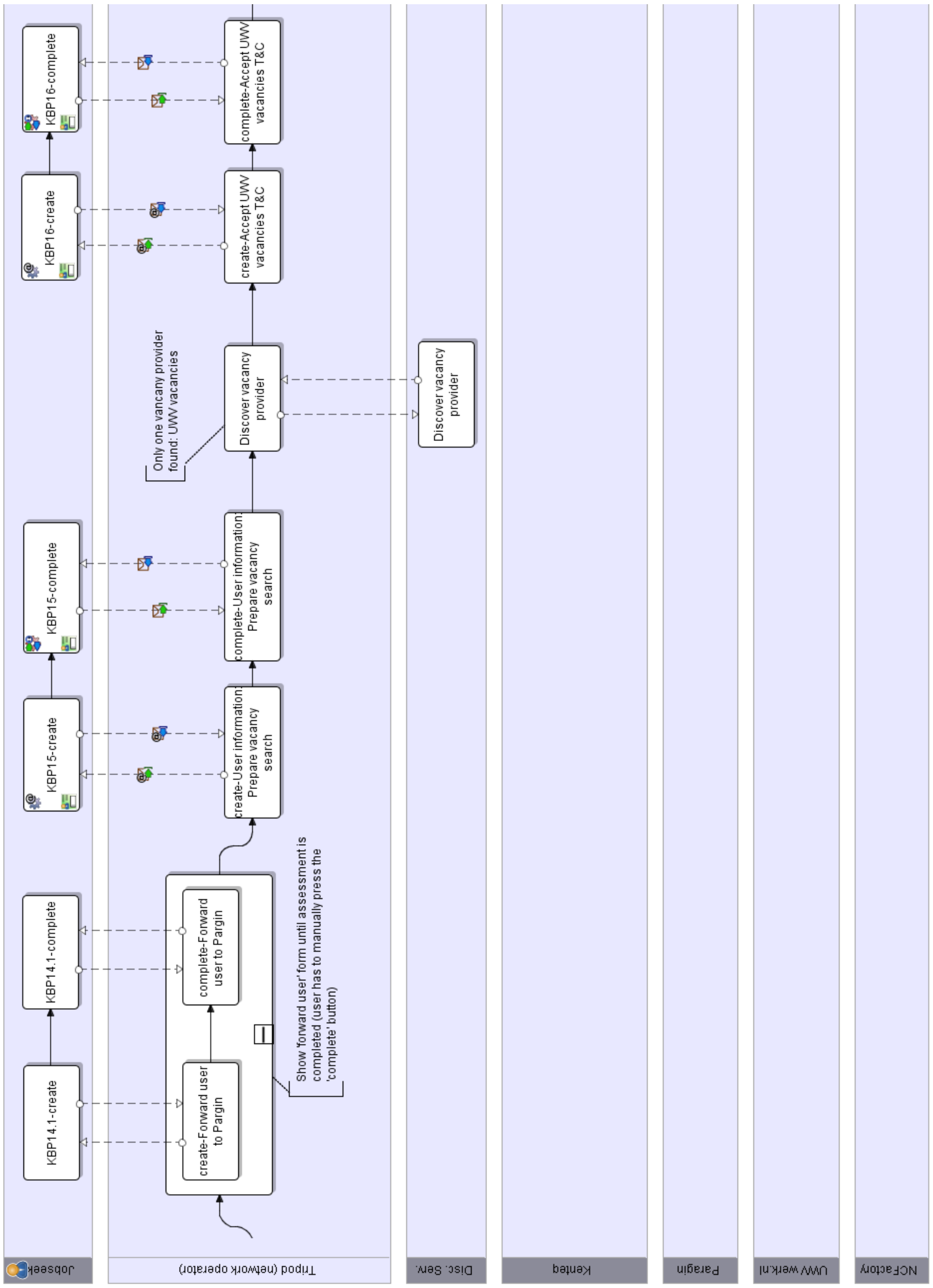

Jobseeker
Tripod (network operator)
Disc. Serv.
Kenteq
Paragin
UWV werk.nl
NCFactory
KBP14.1-create
KBP14.1-complete
create-Forward user to Pargin
complete-Forward user to Pargin
Show 'forward user' form until assessment is completed (user has to manually press the 'complete' button)
KBP15-create
KBP15-complete
create-User information Prepare vacancy search
complete-User information Prepare vacancy search
Discover vacancy provider
Discover vacancy provider
Only one vancany provider found: UWV vacancies
KBP16-create
KBP16-complete
create-Accept UWV vacancies T&C
complete-Accept UWV vacancies T&C

Figure 4.16: Kenteq Employability Process Model – 9

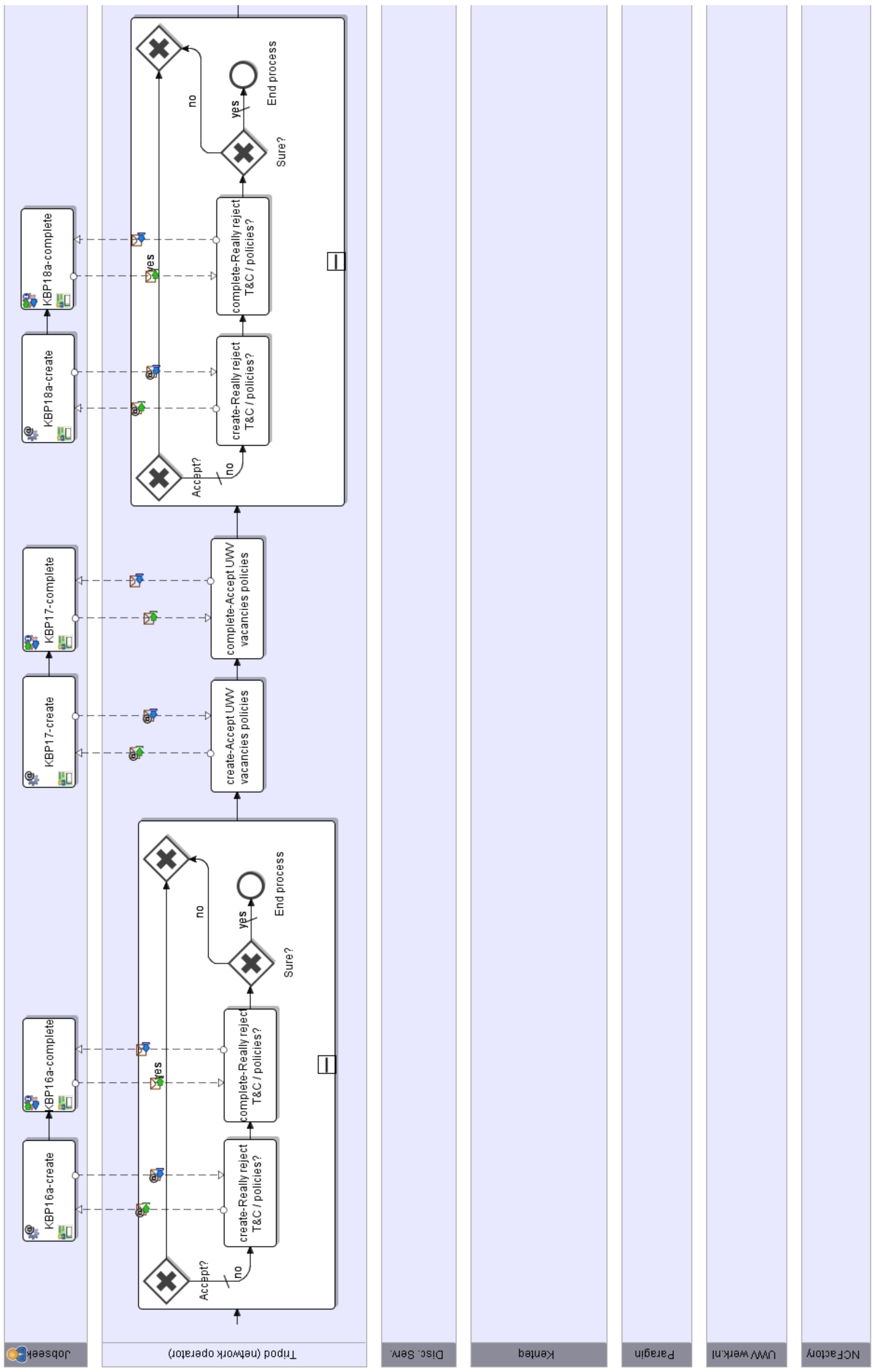

Jobseeker
Tripod (network operator)
Disc. Serv.
Kenteq
Paragin
UWV werk.nl
NCFactory
KBP16a-create
KBP16a-complete
Accept?
no
yes
create-Really reject T&C / policies?
complete-Really reject T&C / policies?
Sure?
yes
no
End process
KBP17-create
KBP17-complete
create-Accept UWV vacancies policies
complete-Accept UWV vacancies policies
KBP18a-create
KBP18a-complete
Accept?
no
yes
create-Really reject T&C / policies?
complete-Really reject T&C / policies?
Sure?
yes
no
End process

Figure 4.17: Kenteq Employability Process Model – 10

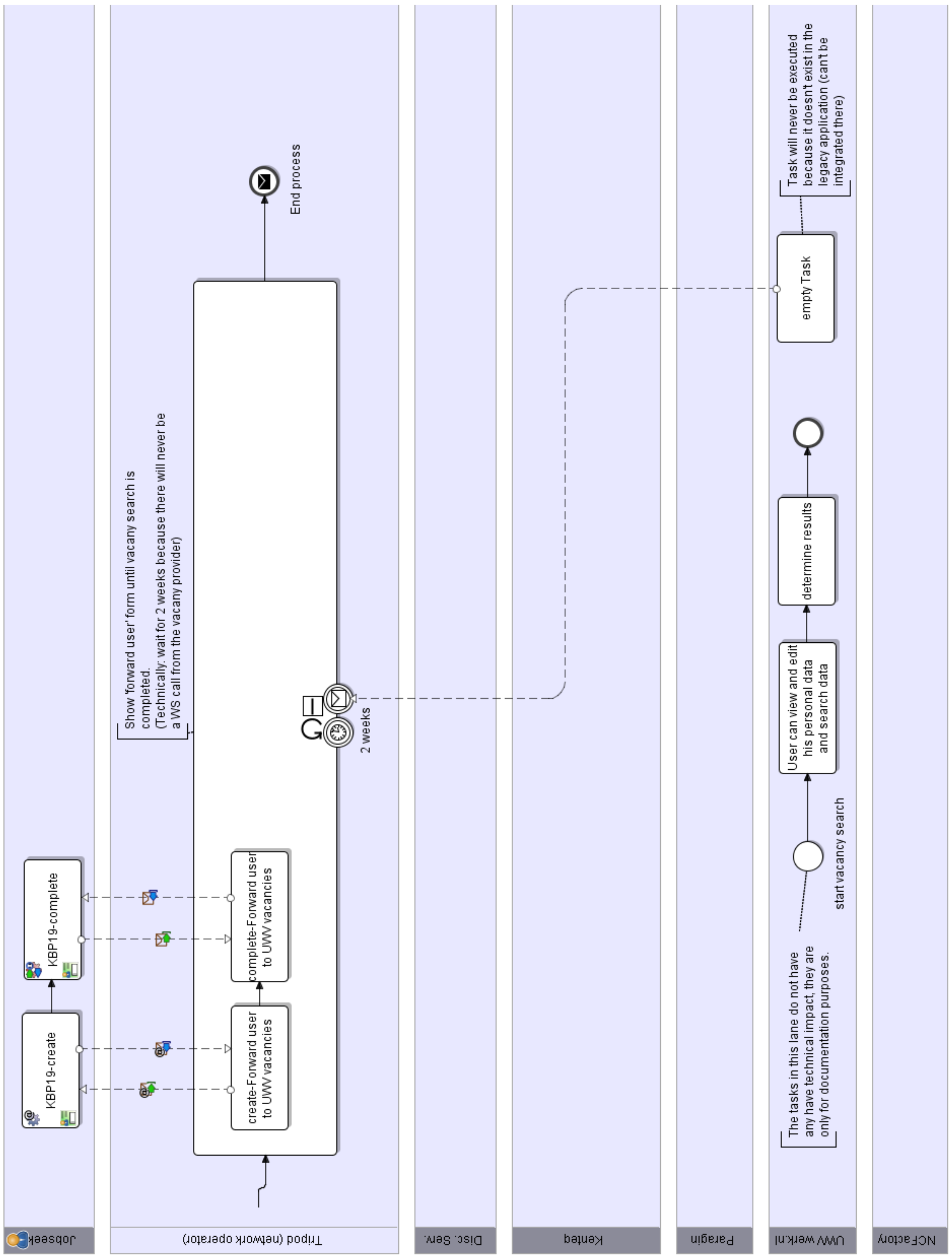

Jobseek
KBP19-create
KBP19-complete
Tripod (network operator)
create-Forward user to UWV vacancies
complete-Forward user to UWV vacancies
Show 'forward user' form until vacany search is completed.
(Technically: wait for 2 weeks because there will never be a WS call from the vacany provider)
2 weeks
End process
Disc. Serv.
Kenteq
Paragin
UWV werk.nl
The tasks in this lane do not have any have technical impact, they are only for documentation purposes.
start vacancy search
User can view and edit his personal data and search data
determine results
empty Task
Task will never be executed because it doesn't exist in the legacy application (can't be integrated there)
NCFactory

Figure 4.18: Kenteq Employability Process Model – 11

5 Break the Glass Functionality for Business Processes

5.1 Motivation

Business-process designers are able to represent security constraints for BP models by means of our security-annotation language [13]. Such security constraints are, for example, role-based access-control specifications to perform tasks. The implementation of such constraints provides security, as only particular role holders are allowed to perform tasks. The same holds for access to protected data, which is restricted to particular users. But such constraints sometimes are too rigid, and more flexibility is required. This is because in certain situations it is desirable to overcome existing security constraints. To illustrate, in an emergency, other role holders than the intended ones should be able to perform certain tasks or to access protected data. Thus, a trade-off between security on the one hand and flexibility on the other hand needs to be facilitated.

The so-called *Break the Glass (BTG)* principle provides flexibility by overriding access rights for process participants. [14] defines the BTG principle as follows: ”Break-glass allows users to override access-control decisions on demand”. The main idea with the BTG principle is to allow for the change of access rights, but in a controlled way. This means that BP designers specify in advance who, in exceptional cases, will have the access rights a process participant does not have in the regular case. We call the explicit action of a user to ask for exceptional access *BTG action* in the following. The three conditions to perform a BTG action are (1) regular access is not allowed, thus access is denied, (2) BTG access is foreseen, (3) a user makes explicitly demand for this access. Next, obligations are typically associated with BTG actions, i.e., operations that compensate[1] for the security violations. Such obligations are application-dependent.

To illustrate, we now describe situations in the TAS^3 scenario (E-Health and E-Employment) BTG functionality is useful for.

E-Health: In the regular case, only certain physicians are allowed to access the sensitive health-record data of patients. But, for example, if there is a life-threatening situation for a patient, other physicians and medical staff might need access to the data. This can be realized by *Breaking the Glass*, meaning that unauthorized process participants (i.e., the physicians treating patients in a life-threatening situation) access the health-record data in a controlled way. In contrast to regular data access, this exceptional access requires specific operations (i.e., obligations) to compensate for the security violation. A typical obligation resulting from such violations is to log the data access by the system. Another obligation is to send an email to the superior of this physician, to the superior of the emergency unit, or to the data owner. The aim of these obligations is to inform relevant individuals that the physician has accessed the health-record data of a patient.

E-Employment: In the regular case in the ”Accreditation of Prior Learning” (APL) scenario, a student’s CV needs the approval of an assessor before it can be sent to a company for a job application. But because an approval typically takes some time, and most job announcements have some deadlines for applying, it might happen that a student wants to apply without an approval. In this case, a student sends his CV, which is not approved by the assessor, to a company, i.e., the student breaks the glass to send his not-approved CV. This results in the obligation that the assessor at least has to be informed of the unauthorized action. Later, when the approval is issued by the assessor, the updated, approved CV is sent again via email to the company the student has applied to. Here, the time for sending an email depends on the BP context, more precisely when the approval takes place. The two obligations in this example show that obligations can be triggered immediately after breaking the glass (i.e., synchronously) or later (i.e., asynchronously).

These examples show how flexibility for access control can be increased in BPMS by providing BTG functionality. [15] discusses other examples for BTG, such as to handle access of de-facto authorized users who have no access rights due to temporarily incorrect access-control policies. Such incorrect

[1] In the following, we use the term *compensation* for the execution of obligations. As a data access cannot made undone, the data access is at least mitigated to an acceptable degree by a set of correcting actions.

policies occur in practice and might be caused, for example, by a design error.

Several approaches implement BTG principles, such as [14], [15], and [16]. BTG works as follows: A regular access is denied, but a BTG action is foreseen. If a user decides to break the glass, access to the resource is given. One way to realize this is to provide temporary roles with predefined rights which have to be used for emergency cases only. This requires the management of emergency roles, such as notifications to users and maintenance of authorizations after usage. Our approach in turn is to change the access rights of users as long as the glass is broken.

Most existing approaches for BTG do not take BP context into account. Under BP context we understand the following, according to [17]: (1) *Activity context*. This is context relating to the task of a BP. (2) *Associated entities*. These are entities like process participants or data owners that are associated with a process instance. (3) *Execution state*. This is information about the execution of a process instance so far. Currently, BTG functionality embedded into BPMS by exploiting BP context is missing. However, we see the following advantages to do so:

- *Consideration of BP-context-specific conditions for BTG actions:* Within a BP schema, the sequence flow can represent implicitly temporal and causal conditions for BTG actions. For example, a BTG option is only allowed after a particular task has been performed (temporal condition), or if the BP instance performs a particular branch of a gateway (causal condition). Using BP context also enables the specification of more sophisticated conditions. For example, in the APL scenario, a student might only access his CV and send it to a company in advance as part of a BTG action, if the assessor has already approved some particular information (e.g., education, job positions) in this CV. Available BTG approaches cannot represent the dependency that the activity of the assessor to approve information represented in a CV runs in parallel to the tasks of the student. Such a condition to perform a BTG action can be expressed by taking the *execution state* and *variables* of a BP instance into account.

- *Specification of BP context for obligation parameters:* Typical obligations are parameterized. For example, an obligation might say that a process participant who performs an activity in parallel and also accesses the BTG-relevant data must be informed about a BTG action. When isolated BTG functionality is used, a system administrator or a developer must provide these parameters for performing an obligation. In our example, the system administrator or the application developer must provide the email adress of the other process participant. By using BP context information about *associated entities*, e.g., the process participants who currently access the same data, the system might determine the receiver of the email automatically. This requires that BP context information is provided to the system component that triggers obligations.

- *Triggering asynchronous obligations by using BP context:* Synchronous obligations are triggered immediately with the BTG action. Asynchronous obligations are triggered at an absolute (e.g., at 2 o'clock p.m.) or a relative point in time (e.g., two hours later, or when a condition holds). In the APL example above, the *execution state of a BP* determines the point in time when an obligation is triggered (i.e., send the CV again to the employer after the task "approval" is finished). Being able to refer to execution states gives way to asynchronous obligations which depend on the progress of a BP instance.

- *Enforcing security constraints together with BTG functionality:* Security constraints, such as Binding of Duty (BoD) and Separation of Duty (SoD) might be in conflict with a BTG action. For example, a BoD constraint is in conflict with the BTG action, if the user is already assigned, and another user is allowed to do a BTG action. To avoid violations of BoD and SoD constraints, context information about *associated entities* should be taken into account. For example, the resource-allocation mechanisms of the BPMS proposes an alternative process participant to break the glass without violating a BoD constraint. Thus, by exploiting BP context, existing security constraints can be fulfilled in a better way together with BTG functionality.

- *Enforcement of obligations as part of the process:* Existing approaches providing BTG functionality handle obligations as external, black box functions that are called on demand. But the embedding of obligations into the BP is advantageous for the following reasons: First, dependent on the application, obligations can be complex, e.g., they might consist of a sequence of actions. BPMS are well suited to manage them. Second, conditions can be specified for the execution of obligations. Examples of conditions are particular ranges of the city code of a student in the APL scenario (i.e. context about *associated entities*) or an amount of fee a student has to pay at least for an assessor. The process engine evaluates these state variables, and selects the required obligations in the BTG case (e.g., inform individuals concerned about a BTG action).

In summary, combining BP context with BTG functionality has several advantages and makes access-control mechanisms in BPMS more flexible.

5.2 Approach

To implement BTG functionality for BPMS, we have developed new concepts for combining BP context with BP access-control mechanisms and managing obligations in a BPMS. We classify the important aspects according to phases of the BP life cycle [18], namely the *design and analysis* phase, the *configuration* phase, and the *execution* phase. For our purposes, the *evaluation* phase is not relevant. In particular, the following issues are challenging: (1) representation of BTG actions and corresponding obligations for BP models (design and analysis), (2) transformation of these representations into extended BP models and into an access-control language (configuration), (3) execution of access-control decisions for process instances (execution). We now describe our approach to support BTG functionality in a secure BPMS from the design phase to the execution phase.

5.2.1 Design and Analysis Phase

BTG allows to access certain resources (i.e., data) and to perform BP tasks. BTG functionality has to be provided in a controlled way, i.e., it must be specified at design time who shall be able to obtain the BTG rights.

Influenced by [15], we distinguish three different types of users involved in a BTG action: the first type are users who have the right to break the glass. The second type are those who have the right to access a resource if BTG has happened. The third type are users which are allowed to perform obligations for the BTG case. For example, only users having the manager role are allowed to break the glass, and their clerks then have access because their manager has activated the BTG status. In practice, it is possible that process participants have both rights. For example, the physician looking after a comatose patient should have the rights to break the glass and to access this patient's health record, for which he has no access rights in the regular case. The third type of users are those who have the right to ”repair” the glass. ”Repair right”, according to our understanding, is the right to compensate the BTG action by performing obligations. This does not mean that this type of users reset the access rights to their regular state, as motivated in [15]. This is because our approach is to reset access rights by means of obligations for a BTG action. It is the task of a process designer to specify when such a obligation has to take place.

A straightforward way is to embed any possible BTG options as a branch in the BP model, treating the BTG case as a possible variation of the regular case. Process events and gateways can represent BTG options. This is likely to lead to very complex BP models, because a single BTG option consists already of a sequence of tasks, and the corresponding obligations can be complex.

Our approach in turn is to embed BTG options into the BP model [2] by means of BTG-specific security annotations. Process designers using our security annotation language from [13] are able to specify role-based as well as user-based access-control mechanisms. Consequently, the rights to perform tasks can be specified for the regular case. We assume to have authorization constraints available for accessing data in the regular case. This can be either explicitly specified by data access policies (e.g., sticky policies), or can be implicitly derived from permissions to perform tasks. We have been extending this security language with BTG vocabulary for accessing data. We now discuss the requirements for these language extensions.

[2]In line with our security annotation language, we represent BP models in BPMN. But our BTG approach is sufficiently general. It can be applied for other BP representations (e.g., YAWL) as well.

5.2.1.1 Requirements for the specification of BTG options

To achieve the advantages as discussed in Section 5.1, we have identified the following requirements for our BTG vocabulary.

- *Representation of BTG access-control constraints:* Obviously, BTG access control must be specified additionally to access-control restrictions for the regular case.
- *Representation of BP-context-specific conditions to perform BTG:* It must be possible to specify conditions under which *BTG actions* should be allowed in relation to the BP context. We have motivated this by an example of the APL scenario in Section 5.1.
- *Modelling of BP-context-specific conditions to trigger obligations:* In particular, triggering asynchronous *obligations* that rely on BP-context-specific conditions must be possible. Further, it should be possible to represent conditions for *obligation* activation.
- *Specification of obligations:* Our approach is to enforce obligations as part of the process. One way to do so is the modelling of obligations as process activities (e.g., as branches of a BP model). But this is time-consuming and leads to complex process models. Thus, we foresee the representation of complex, frequently required obligations at the language level. This requires a vocabulary to specify obligations. We transform the descriptions into an extended BP model in the next step (i.e., configuration phase).
- *Parameters for obligations:* We need functionality to ensure that BP context specific parameters (e.g., associated entities) can become obligation parameters (e.g., see example in Section 5.1 to specify the receiver of an email).
- *Specification of BTG users:* The BTG vocabulary must distinguish the three different types of users involved in a BTG action.

5.2.1.2 Formalizing BP Context

As we consider BP context important for BTG options, we formalize BP context by introducing functions for elements of BP models that return values on the BP context. We need these functions for annotations of BP models in BPMN. The BPMN standard does not specify how to define and how to check conditions that go further than conditions for gateways.

We specify a set of atomic functions to represent BP-context, as follows:

- Execution time of activities. We introduce functions to determine temporal relations for the execution time for activity instances. The three functions *before(activity)*, *during(activity)* and *after(activity)* return *true* iff they are called before, during or after the execution time of a uniquely identified activity, respectively.
- Process entities. The function *performer(activity)* gives the individual performing an activity instance.
- Process variables. The function *value(variable)* returns the value of a process variable.
- Access to data. The function *accessed-data(activity)* returns the data (set(data-ID)) accessed by an activity instance.
- BTG status. The function *BTG(object)* gives true, if the glass is actually broken for an object.

Simple BP-context constraints have the structure *function-name(activity)=value*. Further, logical operators can combine constraints. To illustrate, one might specify a data object access in parallel with a BTG access to the same data object. We formulate: *during(activity-x) AND during(activity-y) AND accessed-data(activity-x) = accessed-data(activity-y) AND BTG(data-ID).*

A detailed description of the semantics of the operators (e.g., the meaning of equality operators for returned sets, or the handling of loops (i.e., the system performs an activity many times) is beyond the scope of this report. We use those functions to formulate *BP context constraints* for BTG options and obligations in the following.

5.2.2 Specification of BTG options for BP models

We annotate tasks in BP models. The annotation term to specify BTG functionality for tasks we propose is as follows:

≪Assignment: type=”BTG”
BTGActivatorRole=”$rolename” | BTGActivatorUser=”$username”
AuthnBTGActivator-attributes=list(attribute, value) idp=$idp-address”
BTGAccessRole=”$rolename” | BTGAccessUser=”$username”
AuthnBTGAccess-attributes=list(attribute, value) idp=$idp-address”
object=”$objectname”
rights=list(”$rightname”)
condition=”$BP context constraint”
obligations=list(”$obligation-ID”) ≫

We now explain the semantics of this annotation term: We distinguish between two roles. *BTG Activator Role* specifies who is allowed to activate a BTG action, i.e., has the right to break the glass. Roles that may perform actions under BTG conditions and thus have access to data are contained in *BTG Access Role*. In the simplest case, one may specify the same roles for the two different attributes.

Accordingly, we propose to allow for a distinction of individual users. We assume that process modellers require this in exceptional cases only. For example, *BTG Activator User* let us specify a particular participant (i.e., an individual) to have the right to break the glass. Process designers can either specify a *rolename* or a *username*, i.e., ”|” is the logical ”XOR”.

Authentication is important for BTG. To enable its specification for these two different roles (or users), there is the parameter *AuthnBTGActivator-attributes* (and for *BTG Access role* respectively). In line with our *≪Authn: list(attribute, value) idp=$idp-address”≫* specification, one can specify attribute/value-pairs and an IDP.

The parameter *rights* specifies which particular rights hold for a BTG action for an *object*, i.e., the resource. For example, the specification $rights = (''update'')$ allows to update data, or the specification $rights = (''read'')$ to read data.

Condition specifies under which circumstances the BTG action is allowed. Attribute values may be *BP context constraints*, as introduced in Section 5.2.1.2. This allows to, say, formalize that a particular activity has to be executed before a BTG action is allowed.

The following parameters are obligatory for a BTG annotation: *type*, *object*, and *rights*. The other ones are optional. If *BTG Access Role* is not specified, a role holder of the BTG-annotated activity gets the rights. If *BTG Access Role* is specified but *BTG Activator Role* isn’t, the value of *BTG Access Role* is used instead.

As one BTG action can have many obligations, and obligations can be complex, we express them as several separate annotations. In the BTG annotation term, there will be a list of obligation-IDs. Each obligation-ID refers to an obligation annotation.

We now describe the representation of obligations in BP models.

We distinguish between obligation patterns and application-specific obligations. Obligation patterns are tasks or sequences of tasks that are likely to be required in many applications and thus can be re-used. Typical obligation patterns are: send email/sms, log data access, and log parts of process execution. Application-specific obligations are tasks with little potential for re-use, such as the obligation that an assessor has to resend a CV once he has approved it.

We propose to handle obligation patterns in the same way as we deal with our so-called user involvements [13]. This means that we provide a vocabulary to describe obligation patterns and represent these pre-defined patterns as language primitives. The secure BPMS transforms those annotations in the configuration phase. This results in an extended BP model. Consequently, process designer can specify frequently required obligations by means of simple annotations, and transformation and execution will be entirely automatic. The embedding of obligations into the process model has another advantage: As we integrate obligations into the application logic, it is the task of the BP-engine to execute obligations and to cancel BP instance if this is not possible [3]. This requires no additional system component to execute obligations

The annotation term for obligations is as follows:

≪Obligation:
id="$obligation-ID"
pattern="$obligationpattern"
parameter=list(("$parametername",$parametervalue"))
CompensatorRole="$rolename" | compensatorUser="$username"
Compensator-attributes=list((attribute, value)) idp=$idp-address"
compensationCondition="$time" | BP context constraint" ≫

The *id* of the obligation is the reference to the BTG annotation. It must be unique within a process model. The *pattern* parameter specifies which obligation type has to be performed, such as *pattern="SendEmail"* or *pattern="AuditAccess"*. Such simple obligation patterns may already represent a sequence of tasks in the BP model.

The value of *parameter* gives the parameters for the obligation call, in the form of (parametername, parametervalue) pairs. For example, sender, receiver, and subject are parameternames for an obligation to send an email. BP context can determine parameter values. For example, an email is sent to the supervisor of a role holder who has broken the glass. As the BP context tells us who has broken the glass, the receiver of the email (i.e., the supervisor of the role holder) can be identified. A prerequisite is an appropriate role model including relationships between roles.

Using the obligation annotation term, process designers can specify roles that are allowed to "repair" the broken glass, i.e., to perform the compensating obligations, by *Compensator Role*. This is required if humans have to perform the obligation. Again, one can specify either a *rolename* or a *username*. In line with the BTG annotation term, we can also specify *Compensator User*. For this role or user, authorization requirements can be specified (*Compensator-attributes*). If no humans are involved, and the BPMS executes the obligation automatically (e.g., it performs a logging), *Compensator Role* does not have to be specified.

The parameter *compensationCondition* specifies when an obligation is executed. It can be a point of time (absolute or relative) or a BP-context constraint. If this parameter is not specified, the obligation is triggered immediately.

Parameters *id* and *pattern* are obligatory. Parameters *parameter*, *Compensator Role*/ *Compensator User*, *Compensator-attributes*, and *compensationCondition* are optional.

One typical obligation is to set the access rights back to the regular case (i.e., to repair the broken glass). Normally, this should take place immediately after an BTG-enabled data access. However, we gain flexibility through the use of temporal conditions or BP-context constraints. Thus, a process designer can alternatively state that BTG access rights should remain intact for a longer time, e.g., until a particular activity is finished.

Process designers need to realize less frequently used obligations for which no BP fragments are available as Web Services.

We will describe the transformation in Section 5.2.3 and give a concrete example from the E-health domain in Section 5.4.4.

[3]We distinguish between the execution and the control of obligations. It is the task of the BP-PEP to control obligations (see Execution Phase).

5.2.3 Configuration Phase

In the design phase, process designers represent BTG functionality by means of annotations. In line with our security constraints for the regular case [19], the BPMS must transform BTG constraints (i.e., BTG and obligation annotations) into BP model extensions, the access-control policy, and configuration parameters. This requires several extensions of the already existing automatic transformation of annotations of our secure BPMS, namely the Security Modelling and Configuration Component for BP (BP-SMC) [13].

In particular, we are currently realizing the following aspects:

- Specification and storage of BTG actions as process fragment.
- Process-oriented representation and storage of obligations.
- Transformation of BTG and obligation annotations into extended BP models.
- Description of BTG options in BP policy. This requires an extension of the access-control policy by BP-context-specific BTG rules.

Specification and storage of BTG actions as process fragment. A prerequisite for a BTG action is that an access request is rejected. From a process perspective, mapping the meaning of a BTG annotation consists of the following steps (see an example in Figure 5.4 on page 108):

1. First, the BTG condition must be fulfilled.
2. If so, a role holder of *BTG Access Role* has to decide whether he/she asks for a BTG access or not.
3. If he/she does so, the system will ask a role holder of *BTG Activator Role* to break the glass. If *BTG Access Role* is equal to *BTG Activator Role*, this step is not needed.
4. Finally, holders of role *BTG Access Role* are allowed to access the data.

Further security constraints may hold, such as authentications of role holders (see specification of BTG annotation term in Section 5.2.1).

These steps take place whenever a BTG annotation is present and an access request has been rejected. To embed these steps efficiently into BPs, we represent them as one process fragment. We store this process fragment in the same way as we do for our user involvements in a repository. For more details on this approach we refer to the TAS^3 Deliverable 3.1, Section 4.6.4 [5]. The functionality of our repository is described in the TAS^3 Deliverable 3.2, Section 5.4.2.6, "processFragmentsRepository package" and Section 5.4.2.7, "Database model of the business process fragments repository" [6].

The meaning of a BTG option is that the BPMS inserts this process fragment into the BP model in the configuration phase, so that these steps take place automatically at process runtime. We give an example of a BTG annotation and the corresponding transformation of the annotation in Section 5.4.

Process-oriented representation and storage of obligations. In general, there are several ways to represent obligations, such as process-oriented, i.e., in the form of (parts of) BP models, or procedural. Process-oriented means in this context that the BP-engine is responsible for the execution. By procedural we understand that our secure BPMS calls tasks, which are not integrated into the BP, as external functions. Our approach is to represent obligation patterns as process fragments and to embed them into the BP model. This is advantageous, because obligations are represented explicitly and do not appear as black-box functions.

We handle obligation patterns in the same way as BTG annotations and user involvements. We store these patterns in a repository.

To bind obligations to BTG actions, our policies contain links to these obligations. In an XACML architecture, the PEP triggers obligations, if required. In our system architecture, the BP-PEP is responsible to do this, by taking BP-context specific conditions that hold for the obligation into account.

In summary, our approach is to embed steps to perform BTG, namely BTG action and obligations, by inserting them into the BP schema at configuration time. However, this is only needed when BTG actions are triggered, and this is not known before process runtime. A dynamic, BP-context specific trigger of these steps at process runtime (i.e., a process migration) would also be possible, but this is more difficult to realize. This is because the insertion of process fragments would also affect running BP instances, including the usual challenges of BP migration, such as maintaining correctness. Thus, we do not follow this approach at the moment.

Transformation of BTG constraints. Our transformation component must also cover BTG and obligation annotations. Figure 5.1 gives an overview of the transformation. One step is to generate access-control rules with obligations for the policy (see the *descriptive* way in Figure 5.1). This policy contains also BP-context-specific conditions. In another step, the transformation mechanism has to insert for each BTG annotation and each obligation annotation a process fragment into the BP model. This is shown in Figure 5.1 by the *adaptive* way. Figure 5.4 on page 108 features an example of a process fragment to be inserted from the E-health domain.

In the last step, if additional security specifications hold for those inserted process fragments, the transformation mechanism extends the BP security policy accordingly. This is shown in Figure 5.1 by the ”BTG Security Constraints” labelled arrow from the schema extension component to the policy. We give an example later in Section 5.4.4 for authentication constraints.

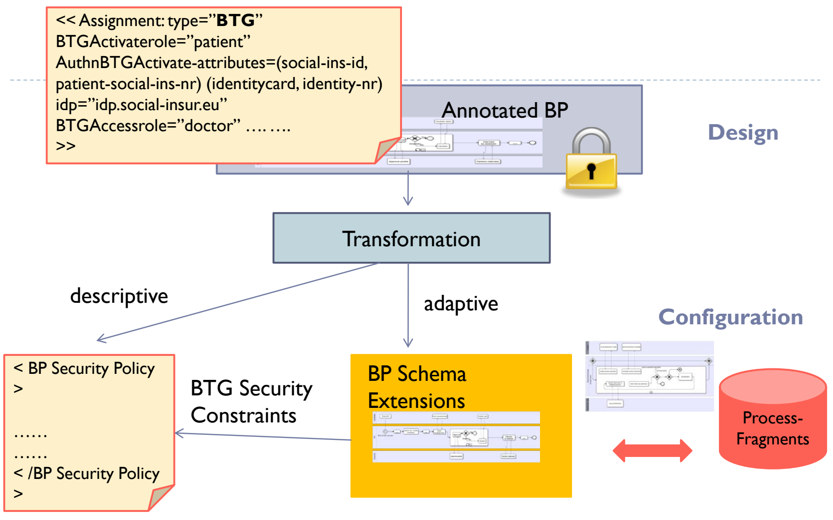

<< Assignment: type="BTG"
BTGActivaterole="patient"
AuthnBTGActivate-attributes=(social-ins-id, patient-social-ins-nr) (identitycard, identity-nr)
idp="idp.social-insur.eu"
BTGAccessrole="doctor" …….
>>
Annotated BP
Design
Transformation
descriptive
adaptive
Configuration
< BP Security Policy >
……
……
< /BP Security Policy >
BTG Security Constraints
BP Schema Extensions
Process-Fragments

Figure 5.1: Overview Transformation of BTG constraints

Description of BTG options in BP policy. We use XACML to represent access-control policies because it is a widely accepted standard. A fundamental concept of XACML is to define access-control policies for so-called targets (i.e., resources). In our BTG context, the XACML targets are data. Policies itself consist of one or several rules. However, the standard XACML architecture does not consider BTG access-control principles. To represent access control for the regular case and the BTG case, a policy should consist of at least two rules: one rule for the regular access-control case, and one rule for the BTG case, which can optionally be refined by a BTG condition. Additionally, we specify obligations within the BTG rules. The XACML standard offers to specify obligations as actions. Obligations must be performed as part of the policy enforcement. In the XACML reference architecture, the PEP (Policy Enforcement Point) component is responsible for the execution of obligations that are related to an authorization decision [20].

As motivated in Section 5.1, we aim to facilitate the binding of BP-context specific conditions to BTG actions. Because the XACML standard does not provide this option, our policy needs to be extended by BP-context-specific aspects. Our enriched policy calls for a PDP that understands these extensions. We discuss this later in Section 5.2.4.

We now describe our extended XACML BP policy for BTG. Because we focus here on BTG, we omit other authorization constraints, such as role holders for activities, or SoD and BoD constraints. The policy consists of two parts. The first part of the policy specifies regular access rights, the second part describes the BTG case.

```
<Rule RuleId="Permission:to:$right:$objectname" Effect="Permit">
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="function:anyURI-regexp-match">
          <AttributeValue
            DataType="string">$objectname</AttributeValue>
          <ResourceAttributeDesignator AttributeId="resource:resource-id"
            DataType="anyURI"/>
        </ResourceMatch>
      </Resource>
    </Resources>
    <Actions>
      <Action>
        <ActionMatch MatchId="function:string-equal">
          <AttributeValue DataType="string">$right</AttributeValue>
          <ActionAttributeDesignator AttributeId="action:action-id"
            DataType="string"/>
        </ActionMatch>
      </Action>
    </Actions>
  </Target>
  <Condition>
    <Apply FunctionId="...">
      ..
    </Apply>
  </Condition>
  <ObligationExpression
    ObligationId="bp:btg:ex:obligation:btg"
    FulfillOn="Deny">
    <AttributeAssignmentExpression
      AttributeId="bp:btg:ex:obligation:btg:attribute:BTGActivatorRole">
      <AttributeValue DataType="string">$BTGActivatorRole</AttributeValue>
    </AttributeAssignmentExpression>
      <AttributeAssignmentExpression
      AttributeId="bp:btg:ex:obligation:btg:attribute:BTGAccessRole">
      <AttributeValue DataType="string">$BTGAccessRole</AttributeValue>
    </AttributeAssignmentExpression>
    <AttributeAssignmentExpression
      AttributeId="bp:btg:ex:obligation:btg:attribute:BTGCondition">
      <AttributeValue DataType="boolean">
        $BTGCondition
      </AttributeValue>
    </AttributeAssignmentExpression>
  </ObligationExpression>
</Rule>
```

This rule states that, in the regular case, the access role $*Regular Access Role* can access the data (in XACML terms target) identified by $*objectname* with the right $\$right$. The latter one is transformed from *list("$rightname")* of the corresponding annotation.

The condition for the regular access case may refer to BP context. To account for this, we introduce a namespace *bp* that contains the BP context. It is specified as a parameter *xmlns:bp="http://tas3.eu/BP-context"* within the XACML *PolicySet* item in the complete policy. In the *Condition* part of the rule, BP context variables can thus be referred to, allowing the user to specify BP-context-specific conditions. Our architecture consists of a PDP component and a context-handler (see Section 5.2.4). The PDP is able to evaluate these conditions at process runtime by asking the context-handler.

To handle the case where regular access is denied, but breaking the glass is possible, we use the XACML obligation functionality. If regular access is denied, the PEP has to ensure that the obligation is performed. This obligation checks whether BTG is possible. To this end, we specify the obligation parameters *BTG Activator Role*, *BTG Access Role*, and *BTG Condition*. These parameters are required to execute the process fragment. If *BTGCondition* holds (e.g., the patient is in emergency health status), the PEP invokes the BTG obligation with the corresponding *BTG Activator Role* and *BTG Access Role*.

When the glass is actually broken, the holder of *BTG Access Role* receives the right to access the data object. This is stated in the second part of the XACML policy, with another rule for the BTG case:

```
<Rule RuleId="Permission:to:BTG:$right:$objectname" Effect="Permit">
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="function:anyURI-regexp-match">
          <AttributeValue DataType="string">
            $objectname</AttributeValue>
          <ResourceAttributeDesignator AttributeId="resource:resource-id"
            DataType="anyURI"/>
        </ResourceMatch>
      </Resource>
    </Resources>
    <Actions>
      <Action>
        <ActionMatch MatchId="function:string-equal">
          <AttributeValue DataType="string">$right</AttributeValue>
          <ActionAttributeDesignator AttributeId="action:action-id"
            DataType="string"/>
        </ActionMatch>
      </Action>
    </Actions>
  </Target>
  <Condition>
    <Apply
      FunctionId="bp:function:glass-is-broken">
      <AttributeDesignator
        MustBePresent="false"
        Category="attribute-category:environment"
        AttributeId="$objectname"
        DataType="boolean"/>
    </Apply>
  </Condition>
  <ObligationExpressions>
    <ObligationExpression
      ObligationId="bp:btg:ex:obligation:01"
      FulfillOn="Permit">
        ...
    </ObligationExpression>
  </ObligationExpressions>
</Rule>
```

This second rule states that *BTGTargetAccess* is possible if the glass has been broken for an object $objectname. To this end, we introduce a function *bp:function:glass-is-broken* that returns for any resource whether its glass has been broken or not. We assume that the PDP can evaluate this *Condition*. If the *Condition* holds, *BTG Access Role* receives the rights listed in *list("$rightname")* in the BTG annotation of object $*objectname*.

The permission to access the data results in obligations different from the regular case. If the BTG access is permitted (*FulfillOn="Permit"*), obligations will be executed. Again, we assume that the PEP passes the information about a BTG permission along and ensures the execution of the corresponding inserted obligation-process fragments. In most cases, the most important obligation is resetting access rights back to the regular case. *When* this should occur can be passed along with a parameter that specifies a BP-context specific condition. We show a BTG-extended policy for an example scenario in Section 5.4.4.

5.2.4 Execution Phase

The starting points to support BTG functionality at execution time are the extended BP model and the BP security policy, both generated in the configuration phase. Both contain BTG-relevant information. The extended BP contains fragments that represent BTG actions and obligations. The BP security policy contains the access restrictions for the regular and the BTG case. In the execution phase, the BPMS executes the BP instances according to the extended model and the BP security policy.

The infrastructure to run process instances is our secure BPMS, developed in TAS^3 [17]. We now discuss features required to provide BTG functionality at runtime. The challenge is to combine BP context with the BTG-enhanced access-control policy and with obligations related to BTG actions. We address the following aspects:

- BTG-capable authorization infrastructure,
- embedding obligations,
- handling BP context.

BTG-capable authorization infrastructure. We need a component which performs authorization decisions according to our BP policy. In the XACML-standard architecture, the task of a PDP is to decide on access-control according to policies. This PDP must be able to handle both, BTG access-control rules as well as BP-context rules. However, a PDP with XACML-standard functionality is not able to handle BTG rules and BP-context rules.

Our TAS^3 partner University of Kent has developed a BTG-PDP [15] that manages BTG authorizations. This BTG-PDP is a stateful PDP, meaning it manages BTG states for resources by responding appropriately. The BTG state of a resource specifies whether or not (i.e., state true or false) a BTG action has happened and thus exceptional access to the resource is possible. We use the BTG-PDP to manage BTG states for data.

But the BTG-PDP is not capable to handle BP-specific-context rules. In our secure BPMS architecture, it is the task of the BP-PDP component to manage BP-specific authorization decisions, namely when process participants get permissions for activities, and to consider BoD and SoD constraints [17]. This asks for a state-based approach to analyze BP instances. The BTG-PDP in turn relies on functionality provided by the BP-context handler (see below).

The main challenge now is to realize generic BTG authorization concepts in combination with BP-context specific ones. The BTG-PDP provides generic BTG decisions, the BP-PDP aims at an authorization infrastructure for BPs, for which BP context has to be taken into account as well. Thus, we make use of both components, the BP-PDP and the BTG-PDP. Our aim is to use the functionality of the BTG-PDP by our secure BPMS.

To provide access-control decisions, [15] proposes a so-called *Master-PDP* that calls a PDP which handles BTG-specific policies. The functionality of this Master-PDP would need to be extended to call a

BP-PDP which is responsible for the handling of BP-context specific policies. However, both the extended Master-PDP, which is able to decide which PDP has to be called, as well as the full functionality of the BP-PDP are currently not available. Further, functionality to combine authorization decisions that affect both components must be provided.

To make use of the features of all components, we proceed as follows: We first implement the functionality to ”pass” BTG-PDP requests within our BP-PEP component. By doing so, we integrate in the first step the BTG-PDP component in our secure BPMS. In the second step, we have to extend the functionality of the context-handler. In the last step, we integrate our BP-PDP so that the BP-PEP is able to handle both BP-PDP and BTG-PDP requests. This requires also an extension of our BP-PDP. The BP-PDP is ”activity-oriented”, i.e., it permits execution rights for activities. Currently, it does not provide access-control decisions for protected data. By realizing this functionality, the access control of the BP-PDP must work for activities and data in a coordinated way. A system with all these features can then provide the full functionality to perform BTG access-control decisions with BP-context specific conditions, as described in Section 5.1.

Embedding obligations. In an XACML-authorization architecture, the PEP has to ensure that obligations are fulfilled [14]. Our approach is to represent BTG as obligations in our BP-policy (see Section 5.2.3). The advantage is that the BP-PEP has to trigger obligations and thus activates BTG automatically if a data request is denied by the PDP. Our PEP in turn has to handle any obligation and has to trigger their execution as part of the BP by calling the BP-engine.

Another task of the BP-PEP is to trigger (i.e., to call) obligations that relate to BTG authorization decisions. To this end, it has to evaluate conditions for obligations first, and it has to pass BP-context variables as parameters for obligations to the BP-engine. It is the task of the BP-engine to execute obligations. To ensure that obligations are executed, it requires that the BP-engine informs the BP-PEP about successful executions.

BP context-handler. The task of the BP context-handler is to provide information about running process instances to the authorization components (BP-PDP, BP-PEP). Having this information, the BP-PDP can evaluate BP-context-specific constraints that refer to BTG annotations and obligations. The context-handler extends our existing BP-PIP (BP Policy Information Point) component by this functionality. It must be able to resolve the functions referring to BP context (see Section 5.2.1). The functionality of the BP context-handler is currently only partially available. For example, the existing BP-PIP does not manage information about access to data, but provides information about execution states of activities.

5.3 Related Work

5.3.1 BTG and Business Processes

For the integration of *Break the Glass (BTG)* principles into business processes, work from two general research directions is of particular relevance. First, work regarding the *Break the Glass (BTG)* principle focuses on mechanisms for overriding of access rights. However, these approaches are usually not directly applicable to BPs due to their static environments. Second, security-related research for BPs considers *context-specific authorization* principles, but is limited in functionality for supporting BTG scenarios.

BTG principles. Several approaches realize BTG principles by overriding access rights without the full integration into a BPMS, such as [14], [15], [16], and [21]. Hereby, the access-control policy plays a central role, because it forms the body of rules for access control. The languages XACML [22] and Permis [23] are two candidates to represent access-control policies. However, both languages are generic, i.e., they do not account for exceptional cases, such as for BTG. To support unpredictable access needs, [21] specify a discretionary overriding of access control in XACML. The authors use obligations to fulfill conditions bound to the access-control overriding. The work is based on the XACML 2.0 standard. As this standard does not handle the combination of obligations, they propose an algorithm to overcome this problem. Our project partners from University of Kent have recently proposed a draft for a BTG profile in XACML [24].

Our TAS^3 partner University of Kent has developed a BTG-PDP [15] that manages BTG authorizations, and a so-called Obligations Service. It is responsible for the coordination and the enforcement of obligations. This requires the registration of obligations. This Obligations Service assumes that obligations are triggered as external black-box functions. Hence, this service cannot be used for our purposes, because we embed obligations into the BP. To provide access-control decisions, [15] proposes a so-called *Master-PDP* that calls a PDP which handles BTG-specific policies. This Master-PDP would need to be extended to call a PDP which is responsible for the handling of BP-context specific policies. However, both the extended Master-PDP, which decides which PDP has to be called, as well as the full functionality of the BP-PDP currently are not available.

BP-context. Access-control languages for BPs in particular (e.g., [25]) address BP-specific aspects, such as binding and separation of duty. [26] supports in enriched access-control policies by embedding so-called ”conditionals” into XACML policies. The system evaluates these policies using an entended XACML runtime environment. [27] proposes to enhance XACML with continuity features, resulting in a so-called U-XACML policy, but a formal model and analysis are missing.

[28] represents complex clinical workflows in a service-oriented architecture paradigm. The authors propose a Prolog-based PDP along with a predefined set of predicates for policy evaluation. However, this approach does not employ standards, and the language cannot express rules in which the relative order of events is stated.

[16] introduces the SECTET framework along with a domain-specific modeling language. The authors model security requirements as part of a business process and translate them into XACML artefacts. [29] specifies so-called Permission Assignment Constraints (PAC) for UML diagrams, using the predicative language SECTET-PL. It allows the specification of fine-grained data-dependent access permissions based on roles. However, how to represent constraints regarding the execution state of BPs in SECTET is open. [30] proposes the integration of a state machine in the form of a coloured petri net into a PDP, but it remains unclear how the approach employs BP context.

5.4 Using BTG in E-health Scenario

5.4.1 Motivation

Healthcare is a field of application in which the secure handling of personal data is especially important. Hence, extensive legislation and regulation exists. The U.S. *Health Insurance Portability and Accountability Act (HIPAA)* outlines the restriction of ”access to all forms of protected health information”. It especially calls for an ”emergency mode operation plan [...] to enable continuation of critical business processes for protection of the security of electronic protected health information while operating in emergency mode” and an ”emergency access procedure [...] for obtaining necessary electronic protected health information during an emergency.” The *EU Directive 95/46/EC (Data Protection Directive)* similarly calls for protection of ”personal data against accidental or unlawful destruction or accidental loss, alteration, unauthorized disclosure or access.” [31] summarizes legal consequences concerning emergency access to healthcare systems, such as obligations for cleanup after account usage. [32] derives from this legislation a generic list of access-control rules necessary for any compliant access-control policy. These rules may serve as a guideline when mapping legal obligations from the field of healthcare to application-independent obligations such as auditing, or security concepts such as separation of duty (e.g., regarding people who access different kinds of (i.e. medical, social, genetic) data.

Influenced by the legal requirements from the healthcare domain, we now focus on supporting BTG functionality for an E-health scenario. We thereby exemplarily realize legal obligations.

5.4.2 BP model of E-health Scenario

We now illustrate the use of some of our BTG concepts in a simplified E-health scenario. Figure 5.2 shows the BP model for a patient’s visit to a physician. When a patient arrives, he has to check in first. Then the physician checks the availability of patient’s health-record data (activity ”Check health record availability”). A ”diagnosis” activity and a ”prescription drug” activity follow. The physician in turn then should update the patient data according to the medical diagnosis and the prescriptions. Finally, the BP terminates, and the patient leaves.

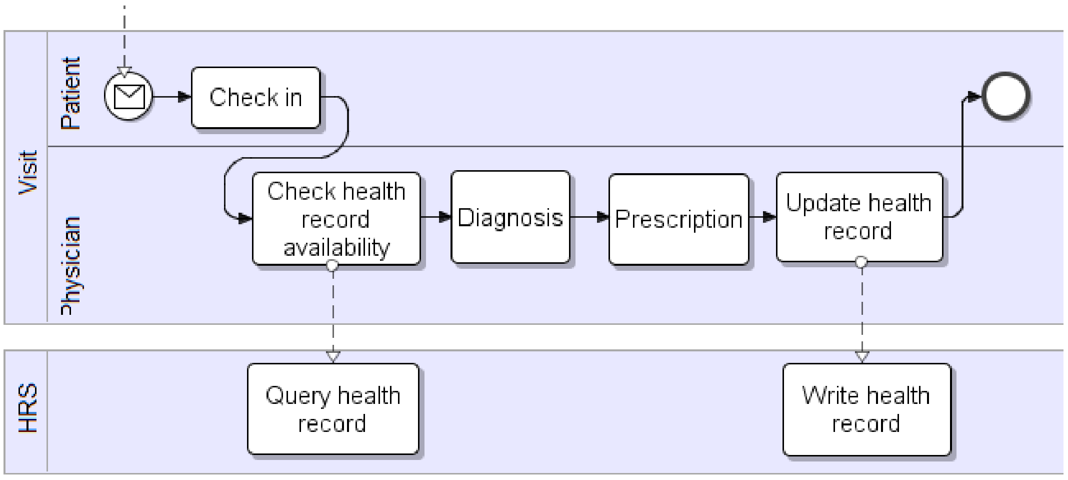


Figure 5.2: Process Model for E-health Scenario

We assume that health-record data are stored externally at a web-service provider (see Figure 5.2 pool "HRS" for health record storage), and these data are sensitive in nature. Thus only authorized and authenticated users should have access to this data. Process designers should handle authorizations restrictively. Using our security-annotation language, we represent the constraints regarding activities of the lane "physician" (see figure) by the following security-annotation terms:

≪Assignment: type="Role" ≫

≪Authn: attributes=(position, medical) (affiliation, eu-medic-approv) idp="idp.med-approv.eu"≫

The assignment term *≪Assignment: ...≫* means that role holders of "physician" are authorized to perform tasks of this lane. The authentication term *≪Authn: ...≫* requires individuals to have the position "medical" (e.g., must have passed an exam) and a European approval to work as medical practitioner. The IDP specified has to handle the approval of these attributes.

We specify the authorization for role holders of patient for lane "patient" as follows.

≪Assignment: type="Role" ≫

Further, the scenario requires an encrypted transmission of health-record data. The annotation

≪Conf: spec="encrypted"≫

for the data flows from the activities of lane "physician" and "HRS" represents this constraint.

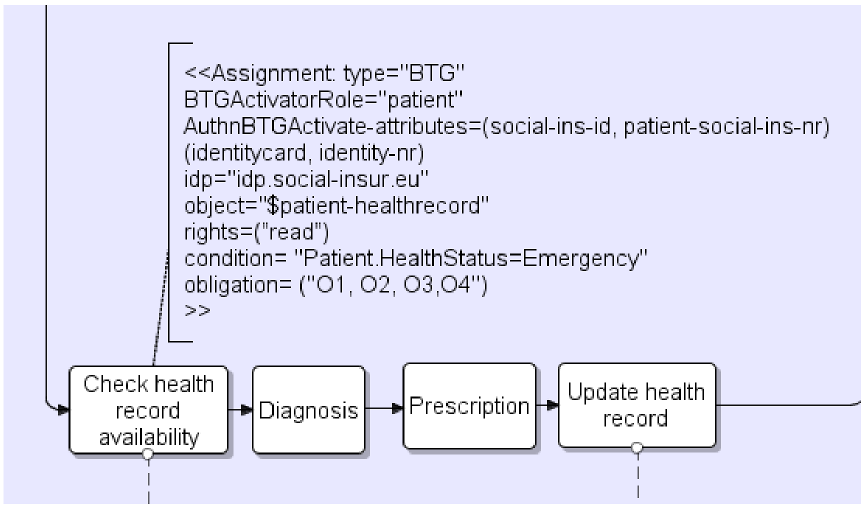
These annotations enforce that only role holders of physician are allowed to perform the activities "Check health record availability" and "Update health record". Both activities ask for access to data. The first one requires a read access, the second one needs a write access. Data access to patient's sensitive health records requires restrictive access-control mechanism by law. If a patient visits his family doctor, we assume that the family doctor is authorized to access his health record. We also assume that the patient is personally known and does not need any authentication, because this has happened once in the past as he had registered as a new patient.

Now suppose that a patient visits another physician. For instance, the patient is on holiday and needs a physician right away. This requires ways to give this other physician access to data. This situation is an emergency. Any physician available must be able to treat the patient and access his patient record in a life-critical situation. This results in a conflict regarding the design of access-control rights: Authorization mechanisms have to restrictively control the access to this sensitive data (i.e., as few individuals as possible should have access), but in exceptional cases access to data needs to be provided to any physician. We represent "exceptional" data-access cases in BP by providing BTG functionality.

5.4.3 BTG Annotations in BP model for E-health

Our aim now is to showcase BTG principles in our E-health scenario, with clear access-control regulations and obligations bound to the data access. This proof-of-concept example can easily be generalized to more complex real-world scenarios.

To enable BTG functionality, we make use of the BTG-annotation term (see Section 5.2.1) for activity ”Check health record availability” and for activity ”Update health record”. Figure 5.3 gives the annotation for the activity ”Check health record availability”. Because the access rights differ for the two activities, we specify two BTG annotations, one for the BTG ”read access”, and one for the BTG ”write access”. A process designer can also specify one BTG annotation for both activities by using the grouping construct. This means that a BTG ”write access” has to be provided for both activities.



```
<<Assignment: type="BTG"
BTGActivatorRole="patient"
AuthnBTGActivate-attributes=(social-ins-id, patient-social-ins-nr)
(identitycard, identity-nr)
idp="idp.social-insur.eu"
object="$patient-healthrecord"
rights=("read")
condition= "Patient.HealthStatus=Emergency"
obligation= ("O1, O2, O3,O4")
>>
```

Figure 5.3: Annotation of Activity ”Check health record availability”

As shown in Figure 5.3, we annotate the activity ”Check health record availability” as follows:

≪Assignment: type=”BTG”
BTGActivatorRole=”patient”
AuthnBTGActivate-attributes=(social-ins-id, patient-social-ins-nr) (identitycard, identity-nr)
idp=”idp.social-insur.eu”
BTGAccessRole=”physician”
object=”$patient-healthrecord”
rights=(”read”)
condition=”value($patient.healthStatus) = EMERGENCY”
obligations=(”O1”, ”O2”, ”O3”,”O4 ”)≫

In our scenario, the patient has to agree to BTG access to his data, i.e., the patient has the role $BTGActivatorRole$. As the BTG agreement is security-relevant, the process designer additionally asks for authentication for the $BTGActivatorRole$. The IDP must authenticate a patient by the $AuthnBTGActivate - attributes$ social insurance number and identity card number. Specifying $BTGAccessRole$ would not be needed in this case (and is also missing in Fig. 5.3), because the $BTGAccessRole$ (i.e., a physician) is equal to the role holder of the activity ”Check health record availability”, due to the role assignment of the lane. We have specified it for illustration purposes only. The objects to be accessed are the patients' health records. We set ”read” access rights for the BTG action.

According to the BTG concept, ”breaking the glass” results in a set of obligations. In our scenario, this means that physician and patient agree to the following compensatory actions for the physician's exceptional health-record access:
O1: The physician's data access is audited so that he can be held responsible regarding the consequences

of the exceptional data access.
O2: At the end of the treatment process, the physician has to write a report about his diagnosis and the corresponding treatment. The physician has to send this report to the family doctor to inform him about the patient's progress.
O3: If, during the "diagnosis" activity, it becomes clear that the patient is suffering from a contagious disease, the doctor has to report this case to the corresponding authorities and the last institutions where the patient has resided. These institutions might differ from those concerned in the regular, non-BTG case (i.e., foreign authorities, hotel instead of workplace).
O4: The access rights to data have to be reset to the regular access case.

Following the terminology from Section 5.1, Obligation O1 serves as an example of synchronous obligations because the auditing begins immediately. O2 is an asynchronous obligation because it refers to a later point in time of the BP (i.e., the end of the treatment activity). O3 is an asynchronous obligation as well, but refers to more complex BP context: The value of the condition that has to be evaluated (i.e., "Does the patient show symptoms that stem from a contagious disease?") is unclear at the point in time of "breaking the glass". Hence, in the BTG case, he might or might not have to write a report to an authority. Additionally, the potential receivers of the disease report are unknown in advance and have to be identified according to the BP context. O4 is essential to the BTG concept, as resetting access rights as soon as possible is crucial to avoid a security breach by provided BTG access rights. Such an obligation will be necessary for all BTG actions concerning data access. To provide flexibility for the application needs, we allow to specify for this obligation *when* the rights should be set back (i.e., at once, or after a certain activity has been executed).

To illustrate, we formulate Obligation O2 using our approach. The other obligations are handled analogously.

≪Obligation:
id="O2"
pattern="send email"
parameter=((from, "$physician.email"),
(to, "$patient.familyDoctor.email"),
(subject,"Patient Report"),
(body,"Find attached a report describing my treatment."),
(attachment,"$physician.ReportFile"))
CompensatorRole="physician"
compensationCondition="after("Update Health Record")" ≫

The example obligation annotation shows that the function *after ("Update Health Record")* represents the asynchronous BP-context condition. Several parameters for the obligation call, such as "from", "to", are listed as parameter name/value pairs. We specify the parameter *compensatorRole* only for illustration purposes. It is not needed here, as the role for the activity annotated is the same as the *Compensator Role*.

5.4.4 Transformation of BTG constraints for E-health

Transformation of BTG annotation. We now illustrate the transformation process for the BTG annotation. Our goal is that the transformation mechanism of our SMC component [13] handles this annotation as follows: It inserts a pre-defined process fragment into the BP model, next to the annotated activity. Our BTG annotation term might contain further security constraints. In our scenario, the IDP must authenticate $BTGActivatorRole$ by particular attributes. The parameters determine additional security constraints that affect the adapted process model. The BP model, extended with the process fragment and the security constraints, is the basis to generate the BP policy in the next step.

Figure 5.4 shows the application logic of the generic process fragment to be inserted (user interactions, interactions to security components, and security annotations are missing). The transformation mechanism extends the BP model by this process fragment (see BP schema extension in Figure 5.1). The process fragment provides two options for health-record access, namely under the regular conditions, e.g., for the family doctor, and for the BTG case. For the BTG case, we distinguish between role holders ”BTG Activator Role”and ”BTG Access Role”, as motivated in Section 5.2.1. The process fragment represent the following steps: First, a physician requests to read the data. Now the BP authorization system may grant or deny the access (first gateway). In the grant case, health-record access is possible. In the deny case, we distinguish whether the BTG condition holds or not (second gateway). If the BTG condition is true (in our case condition= ”$patient.healthStatus = EMERGENCY”), we have to distinguish whether an additional ”BTG Activator Role” has to be asked to give permission or not (third gateway). If the process designer does not distinguish the BTG roles (i.e., ”BTG Activator Role” is equal to ”BTG Access Role”), a role holder of ”BTG Access Role” has to decide to activate BTG or not (activity ”BTG ?”). If he decides to do so, the physician can access health-record data. Otherwise, a role holder of ”BTG Access Role” will be asked to BTG (upper activity ”BTG?”). If he agrees (i.e., ”BTG action wanted”), the role holder of ”BTG Activator Role” is asked to allow to BTG. Only if both role holders have agreed, a BTG Access to the patient record is possible.

A BTG annotation term might contain additional security constraints for the process fragment. Our transformation mechanism has to transform the security constraints by interpreting the parameters of the BTG annotation term in the next step. In our example, we assume that the patient must authenticate himself only in the BTG case. This is because only in this case the patient has a security-relevant task to perform. Consequently, authentications for patients are new security constraints.

We now focus on security constraints that have to be generated for the process fragment, as given in Figure 5.4. Only role holders of ”physician” are allowed to perform the activity ”attempt data access”. This activity ”inherits” the role from the activity ”data access” of the process model. In general, new authorizations and authentications might be required for the roles ”BTG Activator Role” and ”BTG Access Role”. In our scenario, we now require authentications for holders of role ”patient” (i.e., the *BTG Activator Role*). This requirement results in an authentication constraint for role holders ”patient” with the attributes specified above and the IDP. Our mechanism must add this authentication constraint to the BP security policy (left side of Figure 5.1, configuration).

Transformation of Obligations. In the BTG case, the secure BPMS has to execute the corresponding obligations to compensate for the unauthorized data access. We insert obligation-process fragments into the process model in the same way as for the BTG process fragment. This means that we derive for each obligation from the *compensationCondition* the point in time when the corresponding obligation must be executed.

In the case of Obligation O2, the condition *”after(”Update Health Record”)”* specifies that the obligation fragment takes place after the activity ”Update Health Record” (see Figure 5.2). The obligation fragment begins with a gateway checking whether the glass has been broken. If this is the case, an activity is executed that asks the physician to send a report to the family doctor. If a BTG action has not occurred, the obligation is not executed, and ”normal” process execution continues.

Specification of BTG in BP Policy. As described in Section 5.2.3, we extend an XACML policy with BP-context specific constructs. For our current example scenario, the first part of the policy states regular access rights as follows. A physician may access patient data if he is the patient’s family doctor. If this is not the case, the possibility for BTG action has to be checked. We omitted prefixes such as *urn:oasis:names:tc:xacml:1.0* for readability reasons.

```
<Rule RuleId="Permission:to:Read:Health_Record" Effect="Permit">
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="function:anyURI-regexp-match">
          <AttributeValue
            DataType="string">bp:example:patient:record</AttributeValue>
          <ResourceAttributeDesignator AttributeId="resource:resource-id"
            DataType="anyURI"/>
        </ResourceMatch>
      </Resource>
    </Resources>
    <Actions>
      <Action>
        <ActionMatch MatchId="function:string-equal">
          <AttributeValue DataType="string">Read</AttributeValue>
          <ActionAttributeDesignator AttributeId="action:action-id"
            DataType="string"/>
        </ActionMatch>
      </Action>
    </Actions>
  </Target>
  <Condition>
    <Apply FunctionId="function:string-equal">
      <Apply
        FunctionId="function:string-one-and-only">
        <AttributeDesignator
          MustBePresent="false"
          Category="subject-category:access-subject"
          AttributeId="bp:example:physician:physician-id"
          DataType="string"/>
      </Apply>
      <Apply
        FunctionId="function:string-one-and-only">
        <AttributeSelector
          MustBePresent="false"
          Category="attribute-category:environment"
          Path="bp:example/bp:patient/bp:familyDoctorId/text()"
          DataType="string"/>
      </Apply>
    </Apply>
  </Condition>
  <ObligationExpression
    ObligationId="bp:btg:ex:obligation:btg"
    FulfillOn="Deny">
    <AttributeAssignmentExpression
      AttributeId="bp:btg:ex:obligation:btg:attribute:BTGActivatorRole">
      <AttributeValue DataType="string">Patient</AttributeValue>
    </AttributeAssignmentExpression>
      <AttributeAssignmentExpression
      AttributeId="bp:btg:ex:obligation:btg:attribute:BTGAccessRole">
      <AttributeValue DataType="string">Physician</AttributeValue>
    </AttributeAssignmentExpression>
    <AttributeAssignmentExpression
      AttributeId="bp:btg:ex:obligation:btg:attribute:BTGCondition">
      <AttributeValue DataType="boolean">
        bp:example:patient:emergencyhealthstatus
      </AttributeValue>
    </AttributeAssignmentExpression>
  </ObligationExpression>
</Rule>
```

We have a BP-context-specific constraint on the subject which has access for the regular case. This rule states in the *Condition* part that a physician has access to the patient's health record if he is the patient's family doctor. If the access is denied (*FulfillOn="Deny"*), the BTG obligation (i.e., the process fragment) has to be executed. The PEP invokes the BTG obligation with the patient as *BTG Activator Role* and the physician as *BTG Access Role*. BTG can only happen if the *BTG Condition* holds, i.e. there is an emergency.

In the second part of the XACML policy states a rule for the BTG case:

```
<Rule RuleId="Permission:to:BTG:Read:Health_Record" Effect="Permit">
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="function:anyURI-regexp-match">
          <AttributeValue DataType="string">
            bp:example:patient:record</AttributeValue>
          <ResourceAttributeDesignator AttributeId="resource:resource-id"
            DataType="anyURI"/>
        </ResourceMatch>
      </Resource>
    </Resources>
    <Actions>
      <Action>
        <ActionMatch MatchId="function:string-equal">
          <AttributeValue DataType="string">Read</AttributeValue>
          <ActionAttributeDesignator AttributeId="action:action-id"
            DataType="string"/>
        </ActionMatch>
      </Action>
    </Actions>
  </Target>
  <Condition>
    <Apply
      FunctionId="bp:function:glass-is-broken">
      <AttributeDesignator
        MustBePresent="false"
        Category="attribute-category:environment"
        AttributeId="bp:example:patient:record"
        DataType="boolean"/>
    </Apply>
  </Condition>
  <ObligationExpressions>
    <ObligationExpression
      ObligationId="bp:btg:ex:obligation:01"
      FulfillOn="Permit">
        ...
    </ObligationExpression>
    <ObligationExpression
      ObligationId="bp:btg:ex:obligation:02"
      FulfillOn="Permit">
      <AttributeAssignmentExpression
        AttributeId="bp:btg:ex:obligation:02:attribute:compensatorRole">
        <AttributeValue DataType="string">Physician</AttributeValue>
      </AttributeAssignmentExpression>
      <AttributeAssignmentExpression
```

```
        AttributeId="bp:btg:ex:obligation:O2:attribute:compensationCondition">
        <AttributeValue DataType="boolean">
          <Apply
            FunctionId="bp:function:after">
            <AttributeDesignator
              MustBePresent="false"
              Category="attribute-category:environment"
              AttributeId="bp:example:activity:updateHealthRecord"
              DataType="boolean"/>
          </Apply>
        </AttributeValue>
      </AttributeAssignmentExpression>
      <AttributeAssignmentExpression
        AttributeId="bp:btg:ex:obligation:O2:attribute:from">
        <AttributeValue DataType="string">
          bp:example:physician:email</AttributeValue>
      </AttributeAssignmentExpression>
      <AttributeAssignmentExpression
        AttributeId="bp:btg:ex:obligation:O2:attribute:to">
        <AttributeValue DataType="string">
          bp:example:patient:familydoctor:email</AttributeValue>
      </AttributeAssignmentExpression>
      <AttributeAssignmentExpression
        AttributeId="bp:btg:ex:obligation:O2:attribute:subject">
        <AttributeValue DataType="string">
          Patient Report</AttributeValue>
      </AttributeAssignmentExpression>
      <AttributeAssignmentExpression
        AttributeId="bp:btg:ex:obligation:O2:attribute:body">
        <AttributeValue DataType="string">
          I have treated your patient.</AttributeValue>
      </AttributeAssignmentExpression>
      <AttributeAssignmentExpression
        AttributeId="bp:btg:ex:obligation:O2:attribute:attachment">
        <AttributeValue DataType="binaryFile">
          bp:example:physician:report</AttributeValue>
      </AttributeAssignmentExpression>
    </ObligationExpression>
    <ObligationExpression
      ObligationId="bp:btg:ex:obligation:O3"
      FulfillOn="Permit">
      ...
    </ObligationExpression>
    <ObligationExpression
      ObligationId="bp:btg:ex:obligation:O4"
      FulfillOn="Permit">
      ...
    </ObligationExpression>
  </ObligationExpressions>
</Rule>
```

This second rule states that BTG access is possible if the function *bp:function:glass-is-broken* returns *true*, and the physician is able to access the health record at the price of the obligations listed above. We assume that the PDP is able to evaluate this *Condition* part of the rule (i.e., determine whether the glass has been broken for the patient's health record). If the condition holds, the physician receives the *"read"* right for the patient's health record.

If the BTG access is permitted (*FulfillOn="Permit"*), obligations will be executed. Among others, this means that the physician has to send a report to the family doctor with the parameters stated above.

5.5 Evaluations and Limitations

Contributions: In this section we have motivated integrating BTG principles into secure BPMS. We have compiled the advantages of combining BP context with BTG. As breaking the glass and compensating a BTG action by obligations require a sequence of tasks, we manage BTG actions and obligations as integrated parts of processes.

To disburden process designer from representing BTG functionality explicitely, we have proposed a sufficient vocabulary for annotating BTG actions and BTG obligations for the process model. This reduces significantly the specification effort. Our transformation mechanism interprets the annotations and automatically embeds pre-defined process fragments into the BP model. The transformation mechanism also generates a BP policy representing access rules for the regular as well as for the BTG case.

Other parts of our secure BPMS (BP-PEP, BP-PDP) are responsible for authorization decisions at runtime. The BP-engine executes the tasks to break the glass and performs the obligations required. We stress that our work has covered BTG at all levels of process management, from the design phase of BP up to the runtime phase.

We have demonstrated our concepts for an E-health scenario. We first have described the process model and the security annotations, BTG annotations, and compensating obligations. We have shown that using BP context for BTG constraints (i.e., BTG actions and obligations) is advantageous. We have further shown how our transformation mechanism embeds BTG constraints into BP models and into an extended BP policy. We have realized a prototypical implementation for this E-health scenario.

Limitations. From the conceptual part, we have the following limitations: Our BTG approach works for access to data, assuming that data is protected. We do not provide BTG functionality for performing activities of a BP. Our BTG approach is limited to schema extensions at configuration time. We do not support migration of process instances according to BTG needs at runtime. Another issue is the realization of the extended, BTG-specific functionality of the context-handler. The design of the transformation mechanism is challenging, as new requirements come into play by our BTG concepts, such as generating policy rules from process fragments. E.g., authorization constraints of process fragments have to be taken into account, which cannot directly be transformed from the annotation and needs future work.

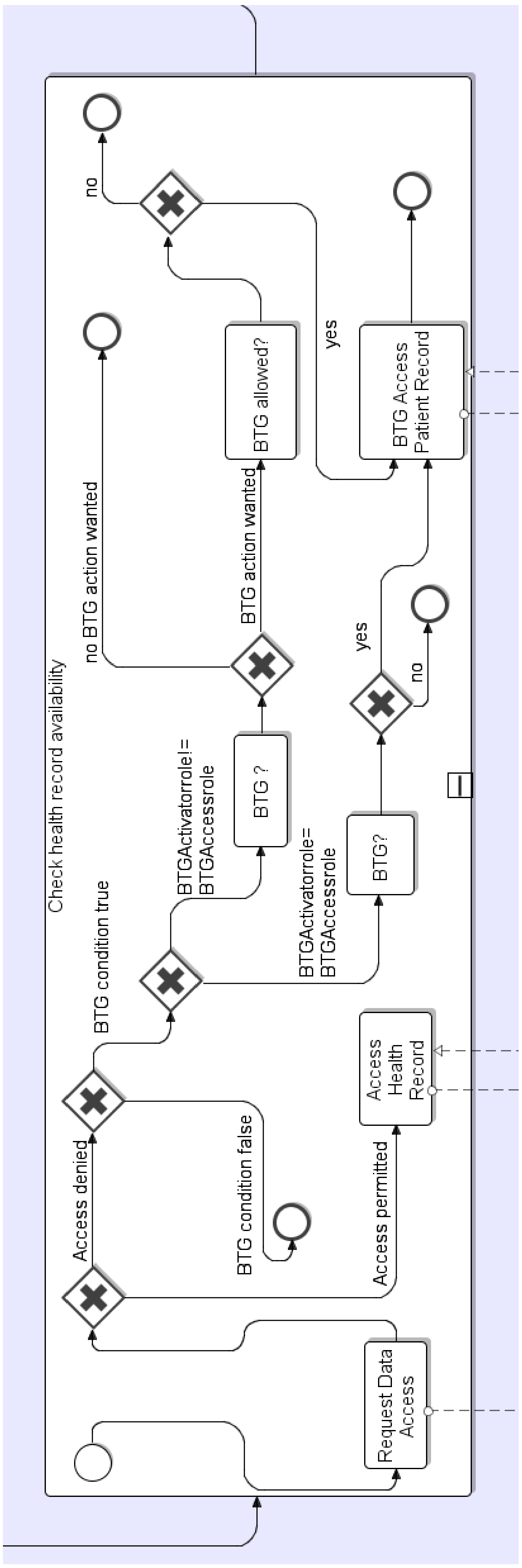


Figure 5.4: Process Fragment for BTG data access

6 Evaluation

In this chapter we describe the evaluation of the use of secure business process technology within the demonstrator scenarios.

6.1 Methodical Approach

For evaluating the demonstrators from the business process perspective, we follow the main topics of WP3 corresponding to the structure of D3.1 (Using Business Process Modelling to Configure Security Components, Developing Security Concepts and Security Policies for Business Processes, and Enforcing Security Constraints by Adapting Business Processes), and analyse step by step how we have realised the security concepts developed in the project.

Criteria of the analysis are:

- Did the specific security aspect appear in the demonstrator scenario?
- How did our solution for this security aspect look like?
- What were the integration challenges?
- How flexible is our solution?
- What was the impact of the demonstrator needs for our topics?

To ease readability and to show the progress from the first demonstrator (Nottingham, as described in Chapter 2) to the demonstrators of the second turn (second Nottingham demonstrator (see Chapter 3) and Kenteq demonstrator (see Chapter 4)), we describe the evaluation of each demonstrator over time.

6.2 Feasibility of the Secure Business Process Concepts within the Demonstrators

The overall challenge for the development of the first Nottingham demonstrator was the first-time integration of the TAS^3 security framework components involved. The business process management assumes the duty to integrate the distributed parts of the application as well as the security components of the TAS^3 security framework. To this end, we have developed the business process model and extended the Intalio|BPMS, a business process management system with rather low security support.[1]

While for the first Nottingham demonstrator, the necessity to build the application on top of a business process management system was not compulsary, the advantages of using a BPMS became very clear during the development of the second demonstrator. The benefit has been obvious regarding flexibility, in particular. During development and even just before finalising, we had to rearrange the sequence and the data flow of the demonstrator quite often to address integration issues and changed functionality of components. These changes would have caused considerable efforts in code rewriting if the application would have been hard-coded. Using a business process management system, we could deal with most of the change requests by rearranging BPMN elements and adapting the data flow in the graphical process model and deploying it again.

6.2.1 Using Business Process Modelling to Configure Security Components

6.2.1.1 Modelling Security on the Business Process Modelling Level

One of the goals of WP3 was to develop a security-modelling language for business processes (see [1]). The demonstrators have contributed in a considerable manner to this goal. Taken the scenarios implemented, we have gathered several new requirements for security constraints to be covered by our

[1] At least in the free Community version which is part of our framework.

security-modelling language. These constraints were not only of particular importance in the context of the demonstrator scenarios but of general relevance for the business process security domain, too. From the first Nottingham demonstrator, for example, we have observed that we need to express that a user selects a service (provider) depending on his/her preferences. This has been a result of discovering services involved in the business process by security and trust properties.

The depiction of the 2nd Nottingham process (Figures 3.5-3.7, on pages 45-47), annotated by security constraints, shows that the security constraints as well as the security-modelling language have been worked out in more detail taking into account the experiences of the first version. During the work on the second demonstrator the security language has been enhanced by constraints such as delegation, more specialised user assignment specifications compared to the version used in the first demonstrator, as well as concepts for user involvement.

The business process for the Kenteq use case of employability made extensive use of one of the user involvement annotations, the *User Consent* annotation (see Section 4.2.2). During the process flow the jobseeker had to agree to the terms and conditions of all involved business partners (namely, TAS^3, Kenteq, a vacancy provider and an e-portfolio provider). The kind of user interaction in each consent giving task was restricted to view the terms and conditions and to agree or not.

The main challenge of the Kenteq demonstrator was the integration of several legacy systems into one TAS^3-enabled application. Besides the already mentioned legal aspects (T&C agreement), this has concerned in particular the way how data exchange between the partners has been implemented. The user had to agree to the data policies of the involved partners, expressed by the *SelectDataPolicy* annotations in the process model. The *Delegation* annotations declare that access rights on personal data are granted to external services.

After March 2011 we continued the evaluation of the security-modelling language with respect to comprehensibility and applicability in an experiment with a large number of participants. One finding from the experiment was that our security language is not limited to the security constraints of the TAS^3 demonstrators but can be applied in completely different scenarios. From the experiment we also gained suggestions for further improvements. The results are described in Section 6.4. Moreover, we have enhanced the language by an additional BTG annotation, which is introduced in detail in Section 5.

6.2.1.2 Knowledge Annotator for Modelling Secure Business Processes

The Knowledge Annotator assists the modeller in applying items of the security-modelling language (see [2], Section 5.1.3.1 for a description of the Knowledge Annotator user client and Chapter 6 for a detailed explanation of the backend). The need for user assistance has been one of the findings we have gained during the development of the first demonstrator. We started developing the Annotator in the second half of 2010. Therefore, we have not had available a stable version, fully integrated into the Intalio|Designer for developing the processes of the demonstrators in the third project year.

During the preparation of the demonstrators for the March 2011 review, the business processes have been developed in close cooperation with the pilot partners, Nottingham and Kenteq, the user interface specialists from the University of Koblenz-Landau, and the usability experts from the City University London (for the last demonstrators). However, the modelling of the business processes has been the duty of Karlsruhe. Because the project partners from Karlsruhe also developed the security language and are familiar with the specification very well, there was no urgent need to use a tool assisting the modeller. Therefore, we have not yet been able to further evaluate the knowledge annotator.

Nevertheless, we are expecting that the ontology-based Knowledge Annotator will help to improve the usability for modellers not so familiar with the BP-security-modelling language. Student assistants having tested the tool without being familiar with the security annotation specification itself support us in our assumption.

The flexibility of our solution is guaranteed by having easy-to-use user interfaces to the ontology backend system, developed by our partners from the Vrije Universiteit Brussel (VUB). These web interfaces allow to make adjustments to the annotation specification. On the technical side we use standard open source technology, which makes the Knowledge Annotator not only run as an Intalio|Designer plugin, but as an add-on to any *Eclipse BPMN Modeler*[2]-based process modelling tool.

6.2.1.3 Transformation of Security Annotations

The security annotations of the business process models provide means to specify BP-specific security, privacy and trust constraints in direct relationship to the BP model. For enforcement, we introduced transformations of the annotations to the execution and enforcement layer, i.e., the secure business process management system. The in-depth specification and implementation of security annotation transformations has been iteratively organized. For the first Nottingham demonstrator, we have implemented the transformation of the user involvement annotations (formerly marked as *RequestToUser*) of the business process model, in a retrospective manner. This first implementation has been rather prototypical and tailored to the specific use case.

Aiming at a more generic solution we have decided to do a complete redesign of the software. For the demonstrators developed during the third project year, we have been able to run more comprehensive transformations. We have performed process model adaptations for certain user involvements, namely *Select Trust Policy* and *Give Trust Feedback*, and have generated a business process security policy comprising all annotated security constraints. The process fragments handling these annotations have been developed during the preparation of these demonstrators.

The third pillar of the transformation process, the automatic configuration of other BP security components has been implemented only in a simple version for the demonstrators developed during the third project year. The configuration parameters provided are those of the concretely used security components.

The Break-the-Glass demonstrator which we have developed as next step, shows how evolving the security annotation items also results in evolving the respective transformations. In particular, there was the need to implement new process fragments for the BTG annotations (see Section 5).

The implementation of the Security Modelling and Configuration tool (SMC) which performs the transformation required quite a huge amount of effort. This tool has to extract the security annotations from the process model, being available in an XML-based format, analyse them, get process fragments from a repository, transform the extended process model according to the extracted annotations, generate a business process policy from some of the annotations, and trigger calls to the BP Policy Information Point (T3-BP-PIP) to pass on configuration data. In a final step the SMC has been integrated into the Eclipse-based modelling tool as a plugin to enhance usability.

The demonstrators allowed us to evaluate the practicability of the transformation to the enforcement level. In some situations, we have found out that transforming security annotations can not be handled in such a straight forward way as we had planned. This necessitated adjustments of the security-modelling language. E.g., it is not trivial to specify audit messages to be triggered by the process. We had to extend the audit annotation by an attribute *policy* which points to an audit policy to be applied when the annotated process event or process task is executed. For example, this policy contains rules like all web service calls and related security actions or all interactions with particular process participants with related security actions must be logged. Another example could be that all activities of a specific role should be logged. On the other hand, the parameterizable *User Consent* annotations in the Kenteq process have been a good example for demonstrating the generic usability of the annotation concept. User Consent annotations hold an attribute *target*, which specifies the object, the user has to consent to. In the Kenteg process the user has to agree to several terms and conditions of involved business partners. Applying User Consent annotations with the attribute *target* pointing to the different terms and conditions, the content of the terms and conditions of all partners could be supposed to be stored in a repository or made retrievable via a URL, and could then be inserted automatically into the appropriate master forms during the transformation.

Our final transformation solution is as flexible as the security annotation language. It can be applied in a great variety of scenarios. The transformation is based on three pillars (process model transformation, BP policy generation, security component configuration), which are themselves based on process modelling standards (i.e. BPMN), standard policy languages (XACML, not fully implemented yet), and standard architectures (PIP, part of the XACML architecture). Therefore the approach can be easily extended by

[2]Eclipse BPMN Modeler website: http://www.eclipse.org/bpmn/

new security constraints or transformation targets. The aspects of flexibility and extensibility have also been considered in the current implementation. (See [2], Section 5.4 (Architecture)).

6.2.2 Developing Security Concepts and Security Policies for Business Processes

6.2.2.1 Federated identity and single sign-on for the user interface

Identity management (IM) is an important element of information system security. IM especially concerns the security aspects authentication and access control. The identity-management system must check the identifiers and attribute values a subject claims to possess, this means authenticating the identity of the subject in a specific context. Based on this knowledge the authentication system can decide whether to grant the subject access to some resource.

The challenge for WP3 was to apply common identity management mechanisms and technologies to the context of a high-level information system in form of a workflow management system. The integration of workflow management systems with identity management systems, esp. with federated identity management (FIM) systems, raises new questions which are discussed in [33].

For the demonstrators we focused on the technological integration of TAS^3 identity management concepts and software, and on the retrieval of business process management specific requirements in matters of (federated) IM.

The integration of TAS^3 identity management concepts comprises the issues of

- integrating single sign-on support into the user interface component of our secure BPMS framework (Business Process CLIENT),
- retrieval, inspection and utilisation of identity attributes, and
- supporting delegation within the TAS^3 security framework.

The SSO enablement of the CLIENT component could already be achieved for the first Nottingham demonstrator (2010 review) by using the SSO functionality of ZXID. For the demonstrators prepared in the third project year we enhanced the integration of ZXID, the Dashboard, and the BPMS-CLIENT component, esp. in terms of user centricity.

During the preparation of the second demo we recognised that setting up a clean integration of the TAS^3 single sign-on mechanisms with the Intalio Tempo software, the basis of the CLIENT component, is not feasible with the available resources. For example the Intalio Tempo software requires business process users to be registered in a file named *security.xml*. This is incompatible with the idea of embedding a BPMS into a federated identity management system, where a user is known beforehand only to his/her IdP(s) and process participants usually get determined not before process instance runtime.

The aspect of identity attribute retrieval and processing was handled in the first Nottingham process and refined in the second demonstrator. In the first Nottingham process the SSO attributes were retrieved from the IdP to decide if the user is allowed to participate in the process and for simplifying the application side registration for him/her. Beyond that the second demonstrator showed that identity attributes can also influence the process flow. Users starting the process with already having set a (preferred) personal data store (PDS) didn't have to register at the application and the application must not trigger the creation of a PDS for them. This behaviour of the application is represented in the process model.

The Kenteq use case includes several service providers participating in the Tripod application. Identity mapping mechanisms provided by ZXID were used to map the identities of the user at different service providers and to make exchange of personal data between service providers possible. Closely associated with the fact of several interoperating service providers is the integration of delegation mechanisms into the scenario as described in section 6.2.2.3.

Beyond that, the Kenteq use case has offered no further challenges towards FIM in the context of business processes.

Regarding the user client side, both applications, the Nottingham placement process as well as the Tripod application, are not typical workflow applications where the user usually picks tasks from a task list. This task list is thought to be part of a load balancing or so-called resource allocation mechanism. In contrast, the Nottingham and the Kenteq use case are mostly single-user or at least one-user centered applications. This key process participant, the *Learner* in the Nottingham process and the *Jobseeker* in the Kenteq process, expects to interact with a user friendly GUI and not beeing confronted with a (at the first glance) confusing list of tasks.

While we had focused on other issues for the first demonstrator, we addressed the aspect of business process user centricity during the development done during the third project period. Using Intalio's chained execution mechanism in combination with an automatic start of the business process we attenuated the mentioned problem. A real world deployment of those processes would require the development of BPMS independent, web service-enabled user interfaces for all human tasks. This is manageable, but requires additional efforts, and has not been a primary goal for the demonstration of secure business-process management and its new mechanisms.

6.2.2.2 Instance-specific user-task assignment and constraints on user-task assignment

One main security aspect for business process management systems (BPMS) is to handle access control within processes and their contexts. This section covers the application of user-assignment concepts in the demonstrators while the succeeding sections deal with permission management on resources and delegation of permissions.

In the first Nottingham process, there was only one user involved per process instance, namely the "Learner". Using security annotations we specified that the Leaner has to be a roleholder of role *student*. Due to the restrictions on process participants, we have not specified any Binding of Duty (BoD) or Separation of Duty (SoD) constraints. Leaving the BoD constraints out appears to be a mistake from our side, because within one proess instance all tasks of the student pool are supposed to be performed by the same person. Nevertheless, we have implemented the required BoD constraint by inserting calls to the PIP into the process model in combination with proprietary Intalio functionality for user-task assignment.

The second Nottingham demonstrator offered more opportunities to evaluate user assignment concepts. The scenario had to deal with two user roles ("Leaner" and "Placement Coordinator"), which have been represented in the business process. The tasks of the Leaner and the tasks to be performed by the placement coordinator had both a BoD constraint attached, expressing that within one process instance the user assigned to each group must be the same for all tasks of this group. We have implemented this constraint in a more integrated manner than for the first demonstrator. The calls to the PIP regarding user-assignment aspects have been routed through the PEP. Lacking a working Business Process PDP and usable interfaces regarding user-assignment handling in the Intalio Tempo component, we had to fall back on integrating proprietary Intalio practices into our BP security framework.

While preparing the demonstrators during the third project period (second Nottingham and Kenteq demonstrator), we came to the conclusion that it is impossible to realise a proper implementation of our user-assignment concepts on the basis of Intalio|BPMS software with reasonable effort. We therefore have started the development of a prototypical user client component for our BP security framework, providing open and well documented interfaces for handling user assignments. This component, together with a Business Process PDP, has allowed us to adequately implement the devised user-assignment features.

The business process of the Kenteq use case of employability has been less complex on the process level than the Nottingham process. Only the course of actions taking place at Kenteq have been explicitly modelled as a business process and have been executed under control of the secure BPMS. The reason is, that the process model includes only one role, namely the "Jobseeker". The BoD constraint has been used to restrict the execution of all tasks belonging to one process instance to a certain user. This feature has been implemented in the same way as in the second Nottingham demonstrator.

An interesting extension of the Kenteq demonstrator would be to realise the application flow for all involved partners as business processes and investigate, if this arises new possibilites for security handling, especially for the user-task assignment aspect. But this was not a main goal of the pilot applications.

6.2.2.3 Delegation

Some application scenarios require the transfer of privileges from one entity to another one. This transfer is called *delegation*. While the first Nottingham demonstrator did not contain any delegation aspects, the second Nottingham scenario allowed us to demonstrate two delegation mechanisms. One mechanism is based on the knowledge of a secret bearer token, and the other is based on attribute delegation[3].

In the application, the learner can choose to let the placement coordinator (PC) select and call a matching service in the name of the learner. To this end, the coordinator needs access permissions to data stored in the learner's Personal Data Store (PDS). From the perspective of the business process, this was technically implemented by passing a secret URL (a form of a bearer token, resulting in data-access rights for the person in possession of it), to the placement coordinator. The business process retrieves the secret URL from the PDS and ensures that it is passed to the coordinator only if the learner has agreed to this delegation. To let the PC perform a call in the name of the learner, the PC needs attributes normally held by the learner. To achieve this attribute transfer, the process calls the TAS^3 Delegation Service and receives a delegation URL. In the next step the coordinator is asked to agree with the delegation. To do so, he opens the delegation URL.

This last step exists only due to technical reasons. The scenario does not require the coordinator to agree to the delegation, because he is responsible for coordinating the placement of the learner. With this, carrying out tasks in the name of the learner can be assumed to be a usual part of his job. So the delegation URL could also be activated automatically. But the PC should still be informed that access rights to the learner's PDS have been delegated to him.

The Kenteq demonstrator comprises several tasks where data access rights have to be delegated to a service provider to handle personal data of the Jobseeker. To reduce the complexity for the Jobseeker, we have chosen a different approach than in the Nottingham scenario. The secure BPMS executes all calls to services requiring access to personal data as identity webservice calls in the name of the appropriate user[4], i.e. the Jobseeker participating in this process instance. The Jobseeker has been informed about this practice in the application's terms and conditions, and, of course, he must agree to them for running the business process.

6.2.3 Enforcing Security Constraints by Adapting Business Processes

Our idea behind adapting business processes is to transform security constraints specified as annotations into security relevant process fragments and integrate these fragments into the existing process model. The process fragments are stored in a master form in a repository and get configured automatically to match the specific requirements of the business process before they get inserted into the appropriate model. Before the first Nottingham demonstrator our plan how to conceptually realise the adaption was still somehow immature. The demonstrator scenario helped us to identify more precisely the "real world" requirements and possible realisation options.

During the development of the second demonstrator our adaptation concept was fully developed, but the implementation, as already mentioned, has been limited to specific examples of process fragments (*Select Trust Policy* and *Give Trust Feedback*). The demonstrator itself has not brought any completely new insights on process adaption. It helped us to define more accurately which parameters are required in the security annotation language and how the process fragments have to look like. A complete redesign of the software, in due consideration of some additional lessons learned from the enhanced Nottingham demonstrator, allowed us to fully realise our adaptation concept.

Regarding the Kenteq demonstrator the implementation of process adaption was not in the focus of our efforts. Nevertheless, it allowed us to make our adaption concept round.

[3]Note that *delegation* is often used in a narrower sense referring only to attribute delegation. We, in contrast, use the term in a broader sense.

[4]See [MuellerBoehm2011] (submitted to a conference) for a full explanation of identity business processes

6.3 Overall Applicability of the Demonstrators

Applying secure BPMS concepts to the pilots resulted in a lot of experiences with the security mechanisms developed so far. We have got valuable insights in the interactions with other security components and functionality of the overall project security framework. These lessons learned provided an important impact on the continous development of our components of a secure BPMS.

The BPMS has been the integration component running in the background during the demonstrations of the e-employability pilots. It has handled calls to security components of the TAS^3 framework, mainly via the BP-PEP. This has required a lot of implementation work, as well as cooperation with many technical partners of TAS^3. A large share of the integration testing, which has been tedious because of the complexity of the various components, has been done by us and was one result of applying our concepts in the pilots.

We have integrated and applied the secure BPM components into three demonstrators in the e-employability domain. The e-health pilot does not make use of business processes. This is because they focus on an application providing access to health data without implementing application-specific process logic by business process technology. However, the results of the e-employability pilots have allowed us to use our security components and enhancements of a secure BPMS in a sufficiently broad manner, which would not be enhanced in a significant way using another application area like e-health. Further, we have analyzed requirements of Break-the-Glass scenarios of the e-health domain, in particular, see Section 5.4.

In order to integrate a further TAS^3 security component, we are focusing on the Break-the-Glass PDP, which has been developed by our partners from University of Kent. Our goal has been to investigate applying BTG principles in BPs. To this end, we started with a systematic analysis of BTG functionality in the context of a BP. We showed advantages for combining BP context with BTG functionality, e.g. specification if BTG is allowed in a certain BP context or as another example, which kind of obligations are possible at which point in time. We have applied our concepts in an e-health scenario. We started from the pure process model and added stepwise security annotations, BTG annotation, and compensating obligations. Thereby we have also shown the advantages of using BP context for BTG constraints (i.e., BTG actions and obligations). We have further shown how the transformation of BTG-annotations can be implemented. This recent application demonstrates how new security properties can be integrated in a secure BPMS using the mechanisms we have developed.

6.4 User Study on Usability of the Security Modelling Language

6.4.1 Motivation

We have presented a rich security-annotation language in [1], [13], and [19]. This language allows to describe classical security aspects for BP needs, such as authorization, authentication, and confidentiality. It also features constructs to let users specify their security- privacy- and trust-related preferences according to the BP context at process runtime. We call the functionality to let users specify their preferences at runtime *user involvements*. Our transformation mechanism generates security policies, adapts process models by inserting process fragments into the process description, and specifies system-component parameters ([13], Section 4.2.3 in [5]). Our extended, secure WfMS ([17], [2]) deploys and executes BP accordingly.

Using our language and transformation infrastructure reduces the effort to specify security characteristics of BP significantly. It also is less susceptible to errors. For example, process designers do not have to care about policy language representations (e.g., XACML) and security implementations. It also provides a very comfortable way for process designers to involve users for specifying their preferences. For example, the selection of a web service according to the specified trust levels considers the availability of appropriate services and requires an interaction with the users if the trust requirements cannot be fulfilled. Process designers can realize such sequences of tasks without any specific security knowledge by generating two annotations.

We have employed this language to enrich business-process models of TAS^3 applications with security constraints (see Section 6.2.1.1). The challenges now are to evaluate the benefit of our language with respect to the following *research questions*: (1) Is our language sufficiently comprehensive, and are users able to correctly annotate on the syntax and the semantic level? (2) Do users understand the meaning of annotations and the benefits of using security constraints at the modelling layer for BPs?

With syntactical and semantic correctness we mean the following: (1) The usage of the correct language construct. (2) The correct specification of the parameters. To illustrate, a ”Role Assignment” requires a type-specification parameter with the value ”role”. We specify a receiver of delegated rights by a parameter of the ”Delegation” term. But a delegation of rights to an activity in a BP that does not need the rights is needless. (3) The correct BPMN element/s to be annotated. For example, a BoD or a SoD for a single activity does not make any sense, if this activity performs only once during a process instance. Overall, generating an annotation term correctly requires some intellectual effort.

But we expect that creating a correct annotation is easier than understanding the meaning. We assume that users are able to explain the annotation terms. Vice versa, they should be able to identify the appropriate annotation term for given security needs that are described in natural language.

Using our approach, process designers can easily specify security constraints and system implementers do not have to care about their realizations. The question here is whether users understand the benefits of our annotation language.

In particular, we have formulated the following *hypotheses*, that form the basis of our evaluation:

- (Hypothesis 1) Individuals can easily learn the annotation language and are able to come up with annotations that are syntactically and semantically correct.
- (Hypothesis 2) Users understand the meaning of annotations.
- (Hypothesis 3) Users are aware of the advantages of our annotation language.

To give answers to these questions, we have carefully performed a user study with more than 200 participants. We have found out that the usability of our language has already been high. Nevertheless, we have marginally improved our language by taking some insights from the user study into account.

6.4.2 Setup of the Study

To have a realistic modelling situation, i.e., the situation process designers typically are in, we have identified the following requirements regarding our setup of the study: (R1) Participants should have knowledge on modelling (data modelling, BP modelling). (R2) Participants must have some basic understanding of security. (R3) Participants should demonstrate their understanding by means of active participation (as opposed to, for example, questions about opinions on usability, etc.). (R4) Participants should work under different application context, namely with sufficient time, and also under pressure of time. (R5) Participants need some incorporation time. (R6) There should be an incentive for participants to contribute good work.

To achieve (R1) and (R2), we have trained the participants on these topics by giving a tutorial (as part of another course) and gave them time for familiarization (e.g., self-study). Further, we have tested the competence of participants by letting them actively practise to model BP and to annotate security constraints (R3). To reflect different application context, we allowed to use any auxiliary background material in one assignment (with sufficient time), but not in the other one (R4). With sufficient time for the assignment, we account on incorporation time for the topics (R5), and later, under pressure of time, we wanted to test whether the participants have tightened knowledge about the annotation language. To achieve R6, we have counted the modelling work as exercise solutions of a course on ”Database Systems”, for which participants earned credit points.

In this study, we have analyzed competence of participants gained in process modelling and their competence, preferences and understanding regarding the annotation language.

6.4.2.1 Phases of the user study

Our user study consists of four phases: tutorials (Phase 1), exercise (Phase 2), questionnaire (Phase 3) and written test (Phase 4). We now describe these phases in detail.

Phase 1: Tutorials: We have taught participants with competence in data modelling in a 100 minutes training unit, which has been part of a course on ”Database systems”. The rationale has been to introduce topics participants need to be familiar with in order to perform subsequent assignments. The topics of this training unit have been: Introduction to BPMS, BP modelling with BPMN, security issues for BP, an overview of our security-annotation language with examples, and an overview on the access control language XACML. The motivation to give an overview on XACML has been that participants should see the complexity of XACML and understand the benefit of our transformation mechanism generating XACML policies automatically.

Phase 2: Perform exercises at home: Participants had to solve an assignment at home within nine days. There were two exercises to do (E1, E2).

With Exercise 1, we had evaluated BP modelling competence of participants, a prerequisite for the subsequent tasks. For Exercise 1, participants had to create a process model in BPMN (either with tool support or handwritten) and had to describe in natural language the meaning of their simple BP. We had chosen a scenario from daily life, namely the visit of a patient to a doctor. Thus, participants did not need any particular domain knowledge. We have asked only for a web service to store health record data. The rationale behind this has been that ”Delegation” and ”Confidentiality” annotations might be possible. We have not asked for particular activities to be present in the BP model. Participants had to model at least four activities, one gateway, and two pools for the BP model. We allowed a maximum of 100 words for the description of the BP. The rationale has been to evaluate whether participants have sufficient knowledge on BP modelling, an important prerequisite to understand and make proper use of our security-annotation approach.

Exercise 2 builds on the result of Exercise 1. We asked participants to annotate their modelled BP with five reasonable, different annotation types. We allowed any annotation terms of our language, the security-annotation terms as well as user involvements. Thus, participants could choose them at their discretion, contingent on how well they fit the BP. Our rationale behind this has been to get many different annotation terms. As students annotated their model they had built themselves, there was sufficient potential to achieve this. Further, participants were asked to briefly explain the meaning of their chosen annotation terms in natural language. We have restricted the description of the five annotations together to 50 words. Exercise 2 contributes to the evaluation of Hypothesis 1.

As auxiliary material to solve these tasks, we have provided the following background material: recorded tutorial material (slides with audio), slides of the tutorial (including literature references), and our technical report on the security-annotation language [13].

After participants had submitted their solutions (more than 200 altogether), we assessed the submitted solutions according to our evaluation criteria. For the BP modelling task (Exercise 1), we evaluated whether the BP models represent a reasonable visit at a doctor and checked whether the BPMN elements required were present. We also evaluated how the descriptions fit the BP models provided. The solutions for Exercise 2 (annotation part) we assessed with respect to the correct syntactical and semantic usage of the annotations. We further evaluated whether the description of the annotation was in line with to the annotation term. In case of contradictions between the annotation and the description (meaning) of an annotation term (e.g., the receiver of a delegation is specified syntactically correct, but described as sender), we reduced the points for the description.

Participants could gain a maximum of 20 points for Exercise 1, and a maximum of 10 points for Exercise 2. The rationale behind this weighting has been to incentivize participants to invest effort to gain the necessary BP modelling knowlege first. We communicated the distribution of points to the participants in advance. For each syntactical and semantic correct annotation of Exercise 2 we gave one point. If the participants described the meaning of the annotation correctly, we gave another point. This leads to a maximum of five points for the annotation terms (E2, Part 1) and a maximum of five points for the description (E2, Part 2).

We gave participants individual written feedback on their performance (number of points, marking of their errors). Further, we gave a short tutorial, aiming to give feedback about typical pitfalls and answered questions about the tasks.

Phase 3: Questionnaire: To get an idea regarding the effort of participants in Phase 2, we let them fill out a questionnaire. The questions were on the effort (in hours) participants invested in solving the exercises (including attending the training unit), which background literature and references they used to solve the exercises, and whether they already had some background knowledge on BP and/or BP modelling.

Phase 4: Written Tests: Finally, security modelling tasks were part of an written test on the training course. We had around 250 participants. Participants had 10 minutes to solve the tasks on security annotations. We decided that participants should have a tough time schedule to solve the assignments.

Regarding the security annotations, there were two tasks to solve. Participants could achieve a maximum of 6 points for Task 1, and a maximum of 4 points for Task 2. We wrote these numbers to the tasks of the test. Task 1 was on specifying security annotations for a simplified, given process model from the university context. This was a simple BP model with six activities, one intermediate event, and three lanes. Task 1 tests the competence for using our annotation language. We use the results of Task 1 to analyze Hypothesis 1. Task 2 was on answering two short questions (T2,Q1, T2,Q2) regarding transfer knowledge and understanding, aiming to analyze the prerequisites and Hypothesis 3.

To solve Task 1, participants had to annotate the BP model according to constraints we have given in natural language. To study participants competence for a broad range of annotations, we divided the participants into two groups with approximately the same level of difficulty (Group A and Group B). We formulated our constraints in natural language, without explicitly stating the annotation term to be used. For example, we described the question aiming for a ”Role assignment” annotation as follows: ”All activities of lane *examiner* must be performed by professors”. Thus, participants first had to exhibit some transfer knowledge, to map the given constraints to annotation terms. To overcome that participants might not have the exact syntax in mind, we provided a sheet with the syntax of our language constructs, but without any description. In total, we had foreseen six minutes to carry out this task (for four annotation terms).

To illustrate, we list the requirements in the following. We presented three constraints to Group A (GA), by asking for: ”Role Assignment” to a lane (T1,Q1,GA), ”Binding of Duty” for two activities (T1,Q2,GA), and two ”User Involvements” for a specification of a trust policy and a web-service selection (T1,Q3,GA). A pitfall here, at least according to our understanding, has been to see the necessity for two annotation terms (”User Involvements”), given one textual description.

Group B (GB) had to solve four constraints, namely that users give consent to terms and conditions by a ”User Involvement” (T1,Q1,GB), an ”Authentication” with some hints for authentication attributes (T1,Q2,GB), the encryption of a message flow (”Confidentiality”)-(T1,Q3,GB), and the ”Delegation” of rights for data to a web service (T1,Q4,GB). The difficulty here (from our perspective) has been to correctly specify the parameters for the ”Delegation” annotation term.

We point out that we have been strict regarding the assignment of marks to solutions: If a participant forgot a parameter required, or if he specified a parameter incorrectly, we reduced the points by 0.5 points per mistake. Consequently, we only gave 0.5 points for identifying the correct annotation term when two parameters (if needed) were wrong.

Task 2 was focused on the meaning and understanding of our language constructs and security in general. Each group had two short questions (T2,Q1, T2,Q2). For each question we calculated two minutes to answer. For group A, we asked to describe briefly for the university scenario the difference of authorization and authentication (T2,Q1,GA). This contributes to prerequisites of the study. The second question was to give two advantages of ”User Involvements”, in their own words (T2,Q2,GA). The latter question contributes to the validation of Hypothesis 3. Group B had to come up with two advantages of representing security aspects in BP models (T2,Q1,GB) and to explain briefly the implications of User Involvements on BP execution (T2,Q2,GB). Both questions of Group B contribute to the validation of Hypothesis 3.

We finally marked the tests of the participants, analyzed the results and provided feedback to those participants who had asked for it.

6.4.3 Results of the Study

Knowledge Required. We had evaluated two prerequisites regarding the knowledge of participants for performing our tasks on the annotation language. We first analyzed whether participants had gained the knowledge to model business processes. Second, participants must have a basic understanding of security.

We studied the first aspect by evaluating the BP-modelling task (Phase 2, Exercise 1). Nearly all participants provided excellent comprehensive BP models and explained the process logic very well. A majority of them had used tools for the graphical representation of the BP model, although this was explicitly not required by us.

According to the questionnaire, participants had invested a lot of effort in the modelling part. We had collected 57 complete, anonymous questionnaires. Participants said to have spent around 1.8 hours on average to familiarize themselves with the topic, 2.22 hours on average for the conceptual part, and 2.20 hours to model the BP and the security annotations. The standard deviation regarding the effort was high. Because the questionnaire was anonymous, we cannot say anything about the correlation between time invested and result quality.

Asking for the material they had used to work off the exercise, all participants answered that they used the tutorial slides. 29 had used documentation on BPMN. 14 participants said to have used documentation from several other sources (internet, books, etc.). Most of the participants answered that they had only little knowledge on BP topics before. We got a mean of 4.13 for the answers on this question (a ”1” means to have good knowledge, a ”5” means to have no previous knowledge).

We see the motivation for this high effort in the relatively high weight (2 out of 3) of the modelling exercise, compared to the annotation exercise. Consequently, the first prerequisite, namely to understand BP models is fulfilled.

We have tested the basic knowledge on security in the written test by asking Group A for the difference of authorization and authentication by using the university scenario as given example (T2,Q1,GA). Table 6.1 summarizes the result for a random sample of 50 out of 100 participants.

Mean	Std.Dev.	very good	good	satisfactory	sufficient
1.24	0.78	40 percent	18 percent	8 percent	66 percent

Table 6.1: Results on Security Knowledge (Phase 4, Task 2, Question 1, Group A)

On average, the participants achieved more than half of the points (mean 1.24, std.dev 0.78 for a maximum of 2 points). 40 percent got the maximum number of points (see column ”very good”), another 18 percent got 1.5 points (see column ”good”), and other 8 percent got 1 point (see column ”satisfactory”). We marked many participants with fewer points because they had not explained ”authorization” and ”authentication” by using examples from the university scenario. Otherwise, without that relatively little transfer knowledge required for this question, the result would have been even better. In sum, there are 66 percent with *sufficient* knowledge (the value of column ”sufficient” gives the sum of the percentage for very good, good and satisfactory). We conclude that participants have sufficient security knowledge in our study to be conclusive. 18 percent of participants had not answered this question at all, but we do not know whether this was due to time restrictions or because of knowledge that has been missing.

We now describe how we have evaluated our three hypotheses.

Hypothesis 1: (The language is easy to learn.) To evaluate this hypothesis, we studied whether participants are able to generate annotations that are syntactically and semantically correct. We therefore analyzed the competence of participants for the annotation task of the exercise at home (Phase 2, Exercise 2, Part 1) as well as their performance in the written test (Phase 4, Task 1).

For the exercise, participants had to choose 5 different annotation terms at their discretion. Our natural expectation has been that participants use the annotations they prefer, and where they are confident to deploy them correctly. Indeed, participants had preferences. They frequently used the annotation terms for ”Role Assignment” (more specifically ”Role Assignment” for lanes), ”Confidentiality”, ”SoD” and ”BoD” constraints. They also had frequently used ”User Involvements” and the ”Delegation” term. Participants used less frequently ”Authentication” and ”User Assignment” annotations. But we were also offered some unexpected annotation terms, such as reasonable ”Mechanism Assignments” for the scenario of the visit to a doctor.

In summary, participants provided good solutions. The results for a sample of 40 participants are as follows: The mean value for the correct annotations is 4.13 points (maximum 5 points), with a standard deviation of 0.92. These good results clearly show that participants are capable of using the annotation language correctly under the given conditions. 19 participants (out of 57) had said in the questionnaire that they had studied our Technical Report for solving the exercise. We ourselves have found this rate surprising.

We also observed some mistakes that turned out to be typical. The specification of ”Delegation” in general and the receiver of the rights of a delegation term in particular, caused some problems. Further, around 20 percent of participants using ”User Involvements” annotations had problems to specify the role parameter in a correct way. Some participants had specified the list of authentication attributes required for the ”Authentication” annotation syntactically incorrect.

Overall, we derived from the questionnaire that participants invested more effort in the modelling of the BP (Exercise 1) than in the annotation language (Exercise 2). In consequence, few participants (in our sample 3 out of 43) had not provided a solution for the annotation terms, but for the modelling task only.

We now focus on the annotation task of the written test (Phase 4, Task 1). Figure 6.2 gives the results for Group A. The values show the mean and the standard deviation for a random sample of 50 out of 100 participants. The first column (Sum Tasks) contains the overall points participants had achieved for Task 1 and Task 2 together. The maximum are 10 points. Column ”Sum Task 1” gives the values for all questions of Task 1 to be solved. This adresses the specifications of annotations from the descriptions given in natural language. The column ”Task 2” summarizes results on the questions that test familiarity with basic security aspects (see results on required knowledge above) and advantages of User Involvements (see Hypothesis 3 in the following).

The other columns on the right hand side show how well participants had come up with the various annotations for Task 1, namely ”Role Assignment” (T1,Q1,GA), ”Binding of Duty (BoD)” (T1,Q2,GA), ”User Involvement” for the specification of a trust policy and ”User Involvement” to select a web service (both for T1,Q3,GA).

	Sum Tasks	Sum Task 1	Role (T1,Q1,GA)	BoD (T1,Q2,GA)	Sel.Serv. (T1,Q3,GA)	Set Trust (T1,Q3,GA)	Task 2 (Q1,Q2,GA)
Mean	5.21	3.15	1.08	0.96	0.38	0.78	2.05
StdDev	2.30	1.60	0.61	0.61	0.59	0.69	1.19

Table 6.2: Results for Group A (Phase 4)

In general, participants had performed well in the written test. The overall mean is 5.21 points for a maximum of 10 points (stddev = 2.30, max = 10). According to our interpretation, this is a positive result, although the mean is only slightly above the half of the maximum points. This is because participants had to solve our tasks under severe pressure of time (only ten minutes). There was only one extremely good participant who achieved the maximum points of ten, and two participants got nine points. This shows that even under harsh conditions it has been possible to make very good contributions.

The participants had solved the ”Role Assignment” (T1,Q1,GA) and ”BoD” annotation part (T1,Q2,GA) of the assignment much better than one on the ”User Involvements” (i.e., ”Select Service” and ”Set Trust Policy”, T1,Q3,GA). One reason might be that the meaning of ”Role Assignment” and ”BoD” is more intuitive than ”User Involvements” for selecting web services and choosing a trust level. Another reason is that there deliberately has been a pitfall, because we had given three questions to solve, but four annotations were necessary to do so. For the third question we asked for two ”User Involvements”, namely to select a service and to set a trust policy. But the fact that two annotation terms were necessary was hidden in the natural language description. Most of the participants had not come up with two annotations for the third task. Only few of them (eight out of 50) had presented two annotations. From these eight participants, only four had come up with an impeccable solution, i.e., they had annotated correctly with the parameters required. This result suggests that our language description could be improved, in particular we could put more effort into the description of relationships between annotation terms, and try out an improved variant of ”User Involvements”.

We now focus on the results for Group B, see Table 6.3.

	Sum Tasks	Sum Task 1	Consent (T1,Q1,GB)	Authn (T1,Q2,GB)	Conf. (T1,Q3,GB)	Delegation (T1,Q4,GB)	Task 2 (Q1,Q2,GB)
Mean	5.21	3.65	0,79	0.99	1.12	0.74	1.61
StdDev	2.10	1.57	0,67	0.49	0.56	0.48	1.22

Table 6.3: Results for Group B (Phase 4)

The overall results are nearly equal to those of Group A (see mean and standard deviation for both groups for all tasks). Here, participants performed better for the annotation task in general (mean of 3.65 points, as opposed to a mean of 3.15 points for Group A). We see the reason in the intricacy of the third question for Group A. Here, it has been obvious that four annotation terms were required. Participants got the most points on average for the ”Confidentiality” annotation (mean of 1.12, std.dev of 0.56). We do not find this surprising, because it is relatively straight-forward. Further, they got around 1 point on average for the ”Authentication” annotation (with std.dev. of 0.49). Creating a ”User Involvement” to give consent leads to 0.79 points on average (std.dev. 0.69). As expected from our observations from Phase 2, the delegation annotation term was the most difficult one (mean of 0.75 points). Overall, we deem the results for the annotation tasks satisfying. They are similar results to the ones of Group A.

We conclude that Hypothesis 1 is fulfilled.

Hypothesis 2: (Participants understand the meaning of annotations.) We analyze this hypothesis in two ways: (a) We evaluate how participants had described the annotations they had chosen themselves (Phase 2, Exercise 2, Part 2). (b) We check whether they are able to map a natural language constraint to the annotation term required (Phase 4, Task 1).

To evaluate the quality of the descriptions of annotation terms, i.e., (a), we look at the results (points) participants have achieved for Exercise 2, Part 2. For the descriptions, the mean is 3.74 points (maximum 5 points) and a standard deviation of 0.98. This result is positive as well, because it means that around 75 percent of the annotations are described correctly. This result is slightly lower than for the annotation part (Exercise 1, Part 1). This confirms our expectations that creating a correct annotation is easier than describing the meaning.

To evaluate how difficult it is to map the description of a security concern to an annotation term, we analyzed the percentage of correct mappings (Phase 4, Task 1). Hereby, we focus on the right selection of the annotation term only and neglect the possibility that parameters might be incorrect.

Within Group A, 62 percent had identified correctly the need for a ”Role Assignment”. 70 percent had correctly inferred from the written text that the constraint is a ”BoD” constraint. We have already discussed the intricacy of the Service Selection with the Trust Policy Specification (see Hypothesis 1). We explicitly gave out this difficult task, which only 16 percent of the participants had solved correctly. There also were some outliers. For example, few participants mixed ”Authorization” with ”Authentication” terms. For Group B, 54 percent had mapped the constraint to a ”User Involvement” to give consent correctly. 52 percent had come up with the ”Authorization” term required, 70 percent with the ”Confidentiality” constraint, given from our natural language description. 72 percent had correctly created for ”Delegation” annotation.

In summary, the results show that individuals are not only capable of using the annotations correctly, they also understand them. This confirms Hypothesis 2.

Hypothesis 3: (The added value of using the annotation language is clear.) In particular, we studied whether participants understand the benefit of our language, as opposed to other approaches for implementing security in BP. We further tried to identify the degree of the understanding of the benefit of ”User

Involvements”. To evaluate this, we look at the answers to the questions regarding transfer knowledge (Phase 4, Task 2) for Group A and Group B.

For both groups, the overall mass performed moderately. Columns ”Task 2” in Table 6.2 and in Table 6.3 show the results for Task 2 for Group A and Group B respectively. The mean of the results is around or lower than the half of the maximum of the points (maximum is 4). A significant number of participants, namely 36 percent for Group A and 32 percent for Group B, obtained zero points for answering the transfer questions.

We now focus on the question regarding the advantages for the ”User Involvements” (Phase 4, Task 2, Question 2) for Group A. Table 6.4 summarizes the results. 14 percent of participants came up with good answers (maximum number of points) and 10 percent with very good ones (1.5 out of 2 points). The last column in Table 6.4 gives the sum of the column ”very good”, ”good” and ”satisfactory”.

Mean	Std.Dev.	very good	good	satisfactory	sufficient
0.77	0.71	14 percent	10 percent	30 percent	54 percent

Table 6.4: Results on Security Knowledge (Group A, Phase 4, Task 2, Question 2)

Table 6.4 shows that the mean for the achieved points is low, namely 0.77 points for a maximum of 2 points. In summary, 54 percent of all participants had achieved a ”sufficient” result, meaning that they had achieved *at least* 1 point.

According to the results, we are not able to confirm Hypothesis 3. We deem that Hypothesis 3 is fulfilled, but our setting does not allow to confirm it. We believe that the results are influenced by three factors: (1) Pressure of time during the written test. (2) Participants had put their effort on the modelling of annotations during the written test. (3) The audience has been confronted with too many new topics.

As we can only speculate about the reasons, we aim to eliminate these aspects by a modified setting of the study and our tuturial in the future. There are several ways that might yield a better understanding of our approach. First, it is important to invest more time during the training phase (e.g., tutorial) to present the annotation approach itself and its advantages. Second, we aim to improve the background material. Third, participants should not be set under extreme time pressure.

6.4.4 Conclusions

In summary, our study has shown that our language is easy to learn and to use. A sufficient number of participants has been capable of generating annotations that are syntactically and semantically correct, and participants performed well regarding the meaning of the annotations. Regarding the understanding of the benefits of the annotation language, we see some room for improvement.

We now discuss the flexibility of our annotation language and the impacts of the study on marginal improvements of the annotation language.

6.4.4.1 Flexibility of the Annotation Language

We have used two application scenarios for this user study, one from daily life (a visit to a doctor) and on from the university context, which is intuitive to participants as well. This shows that our language is flexible when it comes to describing security constraints for scenarios different from those we have investigated in the TAS^3 context. In another study, we had asked participants to develop multiple choice questions on our DBMS course. The motivation was to bring participants to look into the course topics in a way deeper than by studying topics mechanically. In this study we had explicitly asked participants to come up with questions on the course topics that are creative. Some of the participants also proposed questions on the annotation language. To this end, they modelled security constraints for scenarios we never had in mind, such as ordering a pizza or being lost in space and studying a security-annotated BP model to find the way home to earth. This shows that our language is flexible and generic to use.

6.4.4.2 Impact of the Study on the Modelling Language

Our study has identified some typical errors participants made using the security annotation language. To improve usability, we have slightly modified the language. In particular, we have observed that it was difficult for the participants to understand that the ”Delegation” annotation refers to data-access rights (and not to tasks to be delegated) and how to specify such a delegation. Consequently, we have changed the name of the annotation term (now ”D-Delegation” for the delegation of access rights to data) and introduced the owner and receiver parameter names explicitly for ”Delegation”. Additionally, we documented alternative specifications for owners and receivers more clearly. We further improved the syntactical specification for lists of authentication attributes for the ”Authentication” term. Currently, we describe any assignment annotation (e.g., ”Role Assignment”, ”User Assignment”, ”Mechanism Assignment”) by the general ”Assignment” term that requires a type parameter. A more rigid change would be to invest in explicit annotation terms such as ”Role”, ”User”, ”Mechanism” and thereby eliminate the type of the assignment term. This modification would make our language less verbose.

For future work, we are currently investigating in an improvement of our tutorials. We aim to train another group of participants, but with much more knowledge on BPMS. For this group, we want to invest more time for security issues in general and more time on topics for web service selection in distributed environments in particular. The latter contributes to the low percentage of participants who did a correct mapping of the two required annotation terms.

6.5 Visualization of workflow execution and data flow

Personally-identifiable information (PII) is increasingly processed in a distributed way. This makes it much harder for individuals to oversee how their PII is used. In the legal systems of many countries, processing of PII is subject to legal restrictions. In particular, companies have to inform an individual on how they use his PII, and to which external parties they transfer it. We hypothesize that naïve approaches like log messages or plain text are not sufficient to this end. We in turn have developed a user-friendly auditing facility based on business processes (BPs). It visualizes data processing in real time, using the graphical process models one would deploy on a BP engine for execution. We also propose an approach to let a BPMS generate the necessary audit events at runtime. An evaluation of realistic scenarios with users shows that our tool supports them to understand how their PII is used.

We are currently preparing a publication on the visualization tool and its evaluation [34].

7 Conclusions

This report provides a description of three different pilot demonstrators from the employability area. In Deliverable D9.1 the use cases and scenarios have been described from the application point of view. In this report we focus on the business processes involved in the pilots implementation and their security support. The three use cases' descriptions each show particular functionality and concepts of the TAS^3 framework from the perspective of business process support.

The Nottingham use case of Student Placements results in a process with elements which are typical for TAS^3 applications handling with privacy-related data. This demonstrates the usefulness of business processes for integrating humans, data, and services, and additionally providing security mechanisms.

On the basic application level, the use case features the following:

- Handling personal identifiable information, namely the students' e-portfolio for their application to placements.
- Services provided by external entities. Namely, the application uses the matching service which matches the e-portfolio of an applicant with the job offers of the employers.
- Several individuals cooperate within the business process acting in different roles.

On the implementation level, this is facilitated by:

- Persons involved are integrated via so-called human activities, which are realized as web interfaces.
- Service calls are made using web service technology.
- The activities of humans and web services are orchestrated by a business process engine based on a process definition using BPEL.

On top of this, using TAS^3 mechanism provides trust and security:

- Security policies exist for working with the personal identifiable information and for using the matching service.
- Authentication of the process actors is supported via single sign-on and coupling with an identity provider.
- The process execution supports auditing of security-related activities.

Finally, the application itself and its security and trust environment are handled in a dynamic way:

- To support dynamic trust policies, the process includes activities to give feedback of the users on the service quality of the matching service.
- Selecting services is not statically realized in the process design phase, but is dynamically integrated into the process execution. The selection takes security and trust properties into account and uses a discovery service which will be coupled in later versions with the trust and security negotiation service.

The execution of the use case is an integrated application where the application is based on the security and trust framework of the project. The application is realized as a business process, so we have modelled explicitely the activities of the use case and their sequence flow explicitly modelled. A business process execution engine, namely Apache ODE, executes the application process and is coupled with the security and trust services of the framework. Deliverable D3.2 ([4]) describes security components used in the integrated application, which we developed and implemented in Work Package 3 for secure business processes.

The first demonstrator has been a first step for validating the result of the conceptual design of secure business processes. Based on an integrated application, we have demonstrated the use and usefulness of the mechanisms provided in the first version of secure business processes. For that we have been cooperating with the technical partners for integration features with the security and trust architecture, and also with the application partners for application-dependent features of our mechanisms.

The further two demonstrators developed during the third project phase, i.e. the second Nottingham student placement process and the Kenteq mass layoff demonstrator, are described with their functionality, the process model and the component interactions to realise it. The description of the final version of the demonstrators as well as their validation has been added in the last project period.

The second Nottingham use case of student placements is based on the first use case. It additionally involves a personal data store (PDS) and has a focus on an improved user centricity, e.g. providing support for trust and security policy handling and an improved dashboard version.

From the perspective of business processes it provides the following enhanced functionality:

- Refined business process security annotations
- Annotation transformation in business process policy elements and process model adaptations
- Refined assignment of users to tasks
- A business process policy in the demonstrator used for specifying constraints on the following security aspects:
  - Assignment of users to tasks
  - Delegation
- Refined trust and security policy handling
  - Delegation
- Integration of access to a personal data store (PDS) into the business process
- Improved user centricity
  - Refined security modelling
  - Enhanced business process GUIs
  - Automated process start via web service call.
- Improved audit logging

The Kenteq Mass Layoff use case aims to prove that the TAS^3 trust infrastructure can perform in a realistic scenario of several service providers interacting with each other. In particular, the pilot application demonstrates and details the following areas:

- Usability of the general portal
- Single sign-on
- Data discovery
- Multiple Service Providers
- Audit of multiple Service Providers
- Legacy data integration
- Improved Dashboard functionality

During the final working period, we have further validated the security components which we developed for secure business process management in their application in the demonstrators. A new chapter describes the results of the evaluation. For validating our security modelling approach for business processes, we have developed a comprehensive evaluation concept for a large group of participants, and carried out user experiments with different user groups. The experiment and the results are also contained in the Evaluation Chapter. In addition, there is a new application scenario, namely a BTG-application in a business process. This gives new insights on how to leverage business process functionality like process context to the application level to handle BTG-situations and related obligations in a comfortable and enhanced way. Alltogether, we have shown the following findings: first, applications can profit from secure business process management. Second, our approaches significantly improve security specification and enforcement, and, third, we have tightly integrated our security components in the security framework of the whole project.

Bibliography

[1] J. Mülle (ed.), "Design of a semantic underpinned, secure and adaptable process management platform (3)." TAS3 Deliverable 3.1, 3rd Iteration, December 2010.

[2] ——, "Open source software and documentation implementing the design – software for secure business processes." TAS3 Deliverable 3.2, 3rd Iteration, June 2011.

[3] B. Claerhout, S. Winfield, and T. Kirkham, "Pilots specifications and use case scenarios," TAS3 Deliverable 9.1, 2nd Iteration, December 2009.

[4] J. Mülle (ed.), "Open source software and documentation implementing the design – software for secure business processes." TAS3 Deliverable 3.2, 2nd Iteration, June 2010.

[5] ——, "Design of a semantic underpinned, secure and adaptable process management platform." TAS3 Deliverable 3.1, Final Version, October 2011.

[6] ——, "Open source software and documentation implementing the design – software for secure business processes." TAS3 Deliverable 3.2, Final Version, October 2011.

[7] 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/

[8] J. Mülle (ed.), "Design of a semantic underpinned, secure and adaptable process management platform (2)." TAS3 Deliverable 3.1, 2nd Iteration, December 2009.

[9] ——, "Open source software and documentation implementing the design – software for secure business processes." TAS3 Deliverable 3.2, 1st Iteration, December 2009.

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

[11] B. Claerhout, S. Winfield, and T. Kirkham, "Pilots specifications and use case scenarios," TAS3 Deliverable 9.1, 3rd Iteration, December 2010.

[12] *E-portfolio NL*, Nederlandse Norm (NEN) (Dutch national standards body) Dutch national standard (Nederlands Technische Afspraak) NTA 2035, Rev. 2009.

[13] J. Mülle and S. von Stackelberg, "A Security Language for BPMN Process Models." Karlsruhe Institute of Technology (KIT), Germany, Tech. Rep. 2011-09, April 2011. [Online]. Available: http://digbib.ubka.uni-karlsruhe.de/volltexte/1000023041

[14] A. D. Brucker and H. Petritsch, "Extending Access Control Models with Break-glass," in *Proc SACMAT 09, Stresa, Italy*, June 2009.

[15] D. Chadwick (Ed.), "Design of Identity Management, Authentication and Authorization Infrastructure, e, TAS3 Deliverable 7.1, Version 3.0.1," December 2010.

[16] M. Hafner, M. Memon, and M. Alam, "Modeling and enforcing advanced access control policies in healthcare systems with SECTET," in *Models in Software Engineering*, ser. Lecture Notes in Computer Science, H. Giese, Ed. Springer Berlin, Heidelberg, 2008, vol. 5002, pp. 132–144.

[17] 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.

[18] M. Weske, *Business Process Management: Concepts, Languages, Architectures* . Springer, 2007.

[19] 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.

[20] C. A. Ardagna, E. Damiani, S. D. C. di Vimercati, and P. Samariti, *Security, Privacy, and Trust in Modern Data Management*. Springer, 2007, ch. XML Security, pp. 71–86.

[21] B. S. F. Ja'far Alqatawna, Erik Rissanen, "Overriding of access control in xacml," in *POLICY*, 2007, pp. 87–95.

[22] 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

[23] D. W. Chadwick, G. Zhao, S. Otenko, R. Laborde, L. Su, and T.-A. Nguyen, "PERMIS: a modular authorization infrastructure," *Concurrency and Computation: Practice and Experience*, pp. 1341–1357, 2008.

[24] D. Chadwick and S. Lievens, "Break the Glass Profile for XACML V2.0 and V3.0."

[25] Elisa Bertino and Lorenzo Martino and Federica Paci and Anna Squicciarini, *Security for Web Services and Service-Oriented Architectures*. Springer, 2010.

[26] V. Dhankhar, S. Kaushik, and D. Wijesekera, "Securing workflows with xacml, rdf and bpel." in *DBSec*, ser. Lecture Notes in Computer Science, V. Atluri, Ed., vol. 5094. Springer, 2008, pp. 330–345.

[27] M. Colombo, A. Lazouski, F. Martinelli, and P. Mori, "A proposal on enhancing xacml with continuous usage control features," in *Grids, P2P and Services Computing*, F. Desprez, V. Getov, T. Priol, and R. Yahyapour, Eds. Springer US, 2010, pp. 133–146.

[28] J. Mathe, J. Werner, Y. Lee, B. A. Malin, and A. Ledeczi, "Model-based design of clinical information systems," *Methods of Information in Medicine*, vol. 47, no. 5, p. 10, 2008.

[29] M. Alam, M. Hafner, and R. Breu, "A constraint based role based access control in the SECTET a model-driven approach," in *Proceedings of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services*, ser. PST '06, 2006.

[30] B. Katt, M. Hafner, and X. Zhang, "Building a stateful reference monitor with coloured petri nets." in *CollaborateCom*. IEEE, 2009, pp. 1–10. [Online]. Available: http://dblp.uni-trier.de/db/conf/colcom/colcom2009.html#KattHZ09

[31] J. N. Security and P. C. (SPC, "Break-glass: An approach to granting emergency access to healthcare systems. white paper," 2004.

[32] A. Ferreira, "Modelling access control for healthcare information systems:how to control access through policies, human processes and legislation," Ph.D. dissertation, Computing, University of Kent, CT2 7NF, October 2010. [Online]. Available: http://www.cs.kent.ac.uk/pubs/2010/3078

[33] 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).

[34] 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.

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. Standard for modelling business processes in service-oriented frameworks.
- BTG: Break the Glass. Mechanism to override access control decisions on demand.
- Human task: Task in a business process, which contains an interaction with a user. (In contrast to an interaction with a web service.)
- IdP: Identity Provider
- Lane (in BPMN): Used to organise and categorise activities within a pool according to function or role.
- Mahara: Open source e-portfolio system
- PCP: Personal Competency Profile
- PDS: Personal Data Store
- PEP-RQ: Policy Enforcement Point at the Requestor side
- PIP: Policy Information Point
- PIPA: People Initiating Process Activity: A human task used for starting a business process.
- PIP-IR: Instance-Role Policy Information Point
- Pool (in BPMN): Represents a major participant in a process, typically used to separate different organisations.
- SSO: Single Sign-on
- SP: Service Provider
- TN: Trust Network

Amendment History

Ver	Date	Author	Description/Comments
3.0	15.10.2010	Jutta	final revision
2.7	12.10.2011	Silvia, Thorsten, Jutta	address reviewer comments, refine sect.6
2.6	8.10.2011	Jutta	address reviewer comments
2.4	5.10.2011	Silvia, Thorsten, Jutta	address reviewer comments, complete text, actualise BP diagrams
2.3	20.9.2011	Silvia, Jutta	evaluation mod.lang, validation,overview sections
2.2	10.9.2011	Silvia, Thorsten	BTG section, revise validation, setup mod. language validation section
2.1	31.08.2011	Silvia, Thorsten	setup new BTG section and validation section
2.0	29.12.2010	Thorsten	2nd it: Improve diagrams
1.4	23.12.2010	Jutta, Jens, Thorsten, Silvia	2nd it: Revise report and address internal reviewer comments
1.3	14.12.2010	Jutta, Jens, Thorsten	2nd it: Finish for project internal review
1.2	24.11.2010	Jutta, Jens, Thorsten, Silvia	2nd it: Describe 2nd Nottingham and Kenteq demonstrator processes
1.1	10.05.2010	Jutta, Jens, Thorsten	2nd it: Validate 1st Nottigham demonstrator
1.0	04.01.2010	Jutta, Jens, Thorsten	1st it: Refining Impl. description, address internal reviewer comments
0.5	04.01.2010	Jutta	1st it: Introduction, Conclusions
0.4	31.12.2009	Jutta	1st it: Description of Use Case
0.3	22.12.2009	Jens, Thorsten	1st it: Refining diagrams
0.2	15.12.2009	Jens, Thorsten	1st it: Added Nottingham Use Case – diagrams
0.1	06.11.2009	Jutta	1st it: First draft out of blue