UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Rebeka Mändar

UXP Portal 2.0 Functional Requirements Specification

Bachelor's Thesis (9 ECTS)

Supervisor: Margus Freudenthal, PhD
Supervisor: Marlon Dumas, PhD

Tartu 2017

UXP Portal 2.0 Functional Requirements Specification

Abstract:

Cybernetica has developed the Unified eXchange Platform (UXP) — an interoperability platform designed to serve as a secure and reliable data exchange infrastructure. UXP Portal is a component that serves as a universal client application for accessing services over UXP infrastructure. Experience with the initial version of UXP Portal led to the development of version 2.0.

This Thesis describes the process and the outputs of business process modeling and functional requirements specification for the development of UXP Portal 2.0. The development process is based on the Rational Unified Process (RUP). The models were created using Unified Modeling Language (UML) notation. Use-Case Models were developed for both the business and system level domains. The Use-Case Models will serve as an input for the implementation tasks.

The requirements specification process was complicated by the fact that UXP Portal is developed as a product that has no direct customer to elicit requirements from. However, the requirements specification described in this Thesis proved to be sufficient for designing a user interface prototype in cooperation with an external interaction designer.

Keywords: Requirements specification, RUP, UML

CERCS: T120, Systems engineering, computer technology

UXP Portal 2.0 funktsionaalsete nõuete spetsifitseerimine

Lühikokkuvõte:

Cybernetica on välja töötanud toote Unified eXchange Platform (UXP), pakkumaks turvalist ja töökindlat organisatsioonidevahelist andmevahetuskihti. UXP Portal on universaalne klientrakendus üle UXP platvormi pakutavate teenuste tarbimiseks. UXP Portal'i esimese versiooni põhjal tehtud järeldused viisid vajaduseni arendada välja versioon 2.0.

Käesolev bakalaureusetöö kirjeldab UXP Portal 2.0 arendusprotsessi käigus valminud äriprotsesside modelleerimise ja funktsionaalsete nõuete spetsifitseerimise tööprotsessi ja tulemusi. Projekti tarkvaraarendusprotsessi aluseks on Rational Unified Process (RUP). Projekti raames valminud skeemid järgivad unifitseeritud modelleerimiskeele (UML) põhimõtteid. Nii talitluse kui ka süsteemi käitumise kirjeldamiseks on kasutatud kasutusmallimudeleid. Valminud kasutusmallimudelid on sisendiks arendusprotsessi järgnevatele tööülesannetele.

Nõuete spetsifitseerimise muutis keeruliseks tõsiasi, et UXP Portalit arendatakse ettevõtte oma tootena ehk puudub konkreetne klient, kellega koostöös nõudeid välja selgitada. Sellest hoolimata võib välise interaktsioonidisaineriga toimunud koostöö põhjal hinnata, et funktsionaalsete nõuete spetsifikatsioon oli piisava detailsusastmega koostöö alustamiseks.

Võtmesõnad: Nõuete spetsifikatsioon, RUP, UML

CERCS: T120, Süsteemitehnoloogia, arvutitehnoloogia

Contents

Abbreviations

API Application Programming Interface

BUC Business Use Case

CA Certification Authority

CLI Command Line Interface

GA Governing Authority

GUI Graphical User Interface

LDAP Lightweight Directory Access Protocol

OCSP Online Certificate Status Protocol

PSP Portal Service Provider

RUP Rational Unified Process

SaaS Software-as-a-Service

SOAP Simple Object Access Protocol

SQL Structured Query Language

TLS Transport Layer Security

UI User Interface

UML Unified Modeling Language

UX User Experience

UXP Unified eXchange Platform

WSDL Web Services Description Language

XML eXtensible Markup Language

1 Introduction

Cybernetica AS's Unified eXchange Platform (UXP) is a technology that offers a secure means for exchanging information within ecosystems ranging from countries to organizations [1]. Furthermore, the members of UXP infrastructure interacting with each other can vary from big corporations and governmental units to self-employed entrepreneurs and citizens. The information is exchanged via standardized web services but using these services requires developing individual client applications for each service. This process can be technically challenging and costly for smaller organizations. In order to eliminate the step of programming a service client software, Cybernetica has developed a product named UXP Portal that automatically generates service clients based on the service description. This cuts down on the costs and time needed for integrating UXP into organizations' workflows.

In this Thesis, the author describes the requirements specification process and the documents produced for the UXP Portal 2.0 project from the viewpoint of the system analyst. The aim of developing UXP Portal 1.0 was to create a tool for quick and visual testing of UXP compatible services and for sales demonstrations and training purposes. The initial version of UXP Portal was developed into a production-ready tool used in different UXP installations. Gathered feedback showed that addressing the needs of a potentially diverse selection of users demands a new version of the product that is designed with these needs taken into account. The author's task was to specify the requirements for the new version by developing a set of use cases that will be used for implementing the features of the system-to-be. The UXP Portal 2.0 development project follows the Rational Unified Process (RUP) [2] methodology and the documents are completed with Unified Modeling Language (UML) [3] models.

This Thesis has been divided into four chapters. Chapter 2 gives an overview of UXP, lists the main new features in UXP Portal 2.0 and inspects alternatives to the system being developed. In Chapter 3, the author introduces the RUP and its implementation in the UXP Portal 2.0 project. Chapter 4 gives an overview of the business and system domains of UXP Portal 2.0 and includes examples of the artifacts produced in the scope of this Thesis. In Chapter 5, the author discusses the validity of produced results, observed insights and the future of the project.

2 Background

This Chapter gives an overview of the UXP and UXP Portal's position in it. Features of the current version of UXP Portal and the planned features of version 2.0 are listed. Furthermore, the author assesses other universal client applications against the features of the system-to-be.

2.1 UXP

Unified eXchange Platform (UXP) is a product developed by Cybernetica to offer secure and reliable information exchange technology for ecosystems of various dimensions. UXP can be integrated into already existing systems and it supports the growth of the infrastructure by offering seamless linking of new components to existing ones. UXP has a decentralized architecture and the exchanged information is encrypted and signed to ensure confidentiality [1].

UXP and its components have been deployed to serve as a governmental data exchange layer in countries like Namibia [4] and Haiti [5] and the platform has also been used to develop a federated cloud network for a project including Italian and Maltese Ministries of Finance and the UK Police [6, 7]. UXP technology is used by the Estonian X-Road platform to offer e-Government services since 2001 [8] and in Finland since 2015 [9]. The previous two countries are also cooperating in development of X-Road to launch cross-border e-services [10].

2.1.1 UXP Concepts

Each UXP installation is controlled by a Governing Authority (GA) – an organization who has control over both technical and legal aspects of the installation. This includes managing the list of members and defining security policies. An organization has to register as a UXP member to use the UXP infrastructure.

Interaction between UXP members is implemented as service calls. UXP services are web services that correspond to the UXP Message Protocol. This protocol is founded on Simple Object Message Protocol (SOAP) that uses XML as a base for its message format [11]. UXP Message Protocol defines the UXP specific SOAP headers which are used to identify both the service and the service client for a single message transmission.

2.1.2 UXP Components

UXP infrastructure consists of several technical components [1]. The components and their connections are shown in Figure 1.

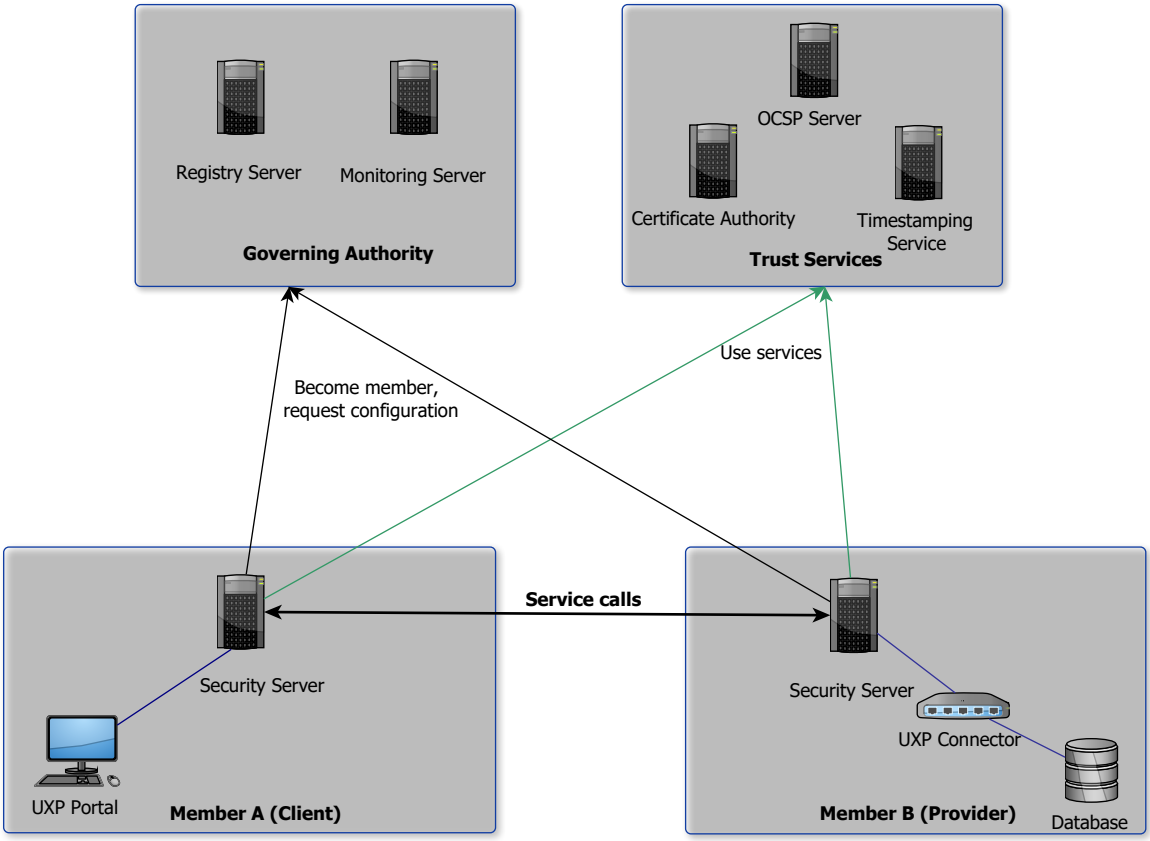

Figure 1. Example of a UXP installation

UXP Registry Server stores information about UXP members, security servers and approved trust services: Certificate Authority and Timestamping Service. This collection of information is distributed to the security servers. UXP Registry Server is controlled by the GA.

UXP Security Server is a gateway between a UXP member's information system and the UXP infrastructure.

Security servers are responsible for routing UXP messages between UXP members. Security servers protect the messages by using digital signatures and encryption. Both incoming and outgoing messages are timestamped and

archived for future verification of data exchange. Furthermore, the security server is used to enforce the access control policy for services which assures that the services are only available to authorized UXP members.

UXP Monitoring Server collects performance information from security servers and statistics about successful and failed requests. This information can be then used to detect issues in the infrastructure, for example bottlenecks and broken services.

Certificate Authority (CA) issues digital certificates for public keys that are used for authentication between security servers and digitally signing the UXP messages.

Timestamping Service issues timestamps required to validate digital signatures in the future.

Online Certificate Status Protocol (OCSP) Service is used to enquire the status of a digital certificate [12].

The implementation of services and client applications is the responsibility of UXP members. The development of the client and service applications can be time-consuming which lengthens the integration process of UXP. Therefore, Cybernetica offers software products that can be used to ease the implementation process [1].

UXP Connector is a web application for implementing and providing UXP-compatible services without any programming. These services use a Structured Query Language (SQL) database as a backend. They are created by defining SQL queries and instructions how to convert between the query result and XML format.

UXP Connector serves as an interface between the security server and the service provider's database. The security server sends the service request to the Connector, where the request is converted to an SQL statement that is then executed on the database. The Connector receives the response, transforms it into UXP message and forwards it to the security server.

UXP Portal is a web application that creates user interfaces for consuming UXP services.

UXP Portal connects to the UXP infrastructure via a security server from where it can request a list of all possible UXP members who provide services in this UXP installation. For each member, a list of available services can be requested. When a service has been identified, the description of the service is imported to the Portal. The service descriptions are written in Web Service Description Language (WSDL). When a user wishes to send a service request, UXP Portal dynamically generates the input form based on the information in the imported WSDL file. After the user has entered the input values, UXP Portal assembles a service call that is forwarded to the security server. The security server routes the service call to the provider and returns the received response to the UXP Portal where it is displayed to the user.

Let's assume that an employee of Member A uses UXP Portal to make a service request which is forwarded to the local security server. The request is sent over a secure channel from the security server of Member A to the security server of Member B. The secure channel is implemented by a mutually authenticated Transport Layer Security (TLS) connection. From the security server of Member B, the request is sent to the UXP Connector, which responds with data received by querying the connected database. The security server of Member B sends the response to the Member A's security server via the secure channel. The receiving security server sends the response to the UXP Portal where it is displayed to the employee who made the request.

2.2 History of UXP Portal

Development of the first version of UXP Portal was driven by the need for a tool to demonstrate consuming UXP services during trainings for service developers and during UXP demonstrations for potential customers. As a result of positive feedback and interest, UXP Portal was developed beyond the prototype stadium by adding the following features to support practical use:

1. *User management*

   Gives the organization means to easily control access to services for each employee.

2. *Organization management*

   Users of one UXP member are grouped under one organization entity in UXP Portal. This offers the opportunity to host multiple UXP members in one UXP Portal installation without the users interfering with each other.

3. *Enforcing TLS on connections*

   UXP security servers support configuring TLS on connections to service clients to assure secure communication especially when the security server and UXP Portal are not located in the same network. In order for the certificate based authentication to function, security settings have to be configured on both ends of the communication channel. In Portal 1.0, this is done via command line interface (CLI).

4. *Service management*

   In order to not search for a service each time a user wishes to send a request, the services available to the organization are registered in the UXP Portal configuration for quick access.

5. *User level service access rights*

   On security servers, the access to use a service is set on organization level. Organization's internal policies might state that some employees are not allowed to access all registered services. Therefore, UXP Portal was designed to allow setting additional access rights within one organization.

Adding new features required separating the functionalities between different user roles. User roles in UXP Portal and their functionality domains are the following:

Portal Administrator manages UXP Portal and the UXP members registered to use one Portal installation. The registered members are called organizations.

User Administrator manages users inside an organization.

Service Administrator manages services and users' rights to access services inside an organization.

Service User consumes UXP services inside an organization.

Communication with the UXP customers and in-house assessment of the system brought out the need to increase the usability and flexibility of UXP Portal. The growing list of both functional and non-functional requirements led to the decision to develop UXP Portal 2.0.

2.3 Requirements for UXP Portal 2.0

2.3.1 Usage Scenarios

The Vision of UXP Portal 2.0 states that the system will be designed to satisfy four main usage scenarios and therefore be used by:

1. small organizations who wish to use UXP services without investing in developing service client applications;
2. bigger organizations who are capable of developing software but need a quick replacement until the applications are implemented;
3. Cybernetica during trainings and demonstrations;
4. service developers who wish to test their UXP services.

As smaller organizations might not have IT capabilities, setting up and managing the UXP Portal on a daily basis should require the least possible effort for non-technical users.

2.3.2 Deployment Scenarios

It is possible to deploy UXP Portal in different ways. The particular deployment choice depends on the IT capability of the service client organization. The deployment scenarios are the following:

1. UXP Portal is deployed within the organization using it for consuming UXP services. This scenario is suitable for larger organizations who also operate their own UXP security server. The scenario is presented in Figure 2.
2. UXP Portal is deployed by a company who intends to offer the UXP Portal as Software-as-a-Service (SaaS). The potential customers for UXP Portal as a service are UXP members who do not have the capabilities or interest

in operating the hardware required to run UXP Portal. See Figure 3 for illustration.

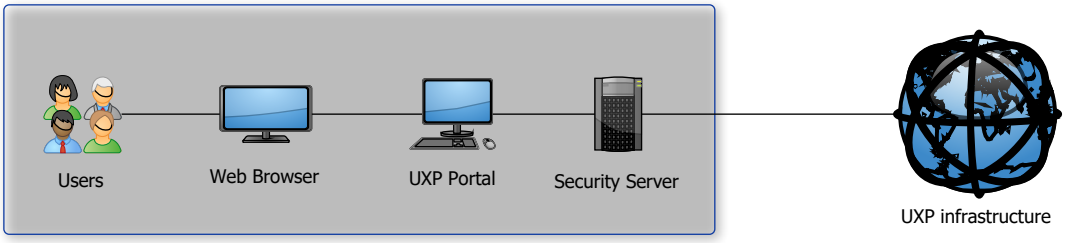

Figure 2. UXP Portal as local installation

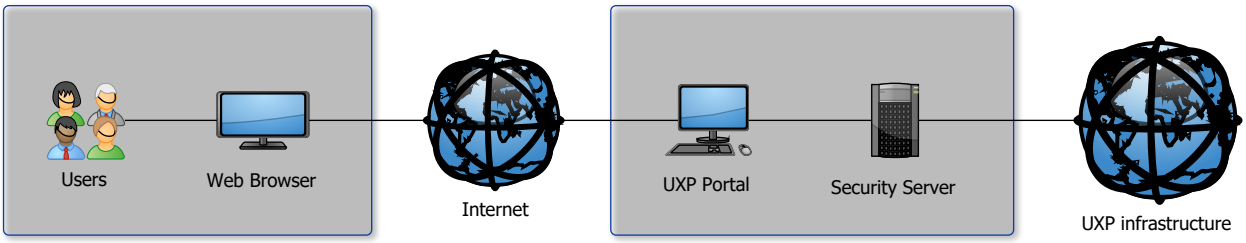

Figure 3. UXP Portal as SaaS

2.3.3 New Features in Portal 2.0

This section describes the features that are required to support the four main usage scenarios listed in Section 2.3.1 and the deployment scenarios in Section 2.3.2. The features already included in the UXP Portal 1.0 (see Section 2.2) still apply as requirements for the new version unless they are overwritten by any of the following requested features.

1. *User management with LDAP integration*

   In the current version of UXP Portal, the User Administrator has to create a new account for every user and the user information is stored in the Portal database. Large organizations with several departments have already gathered their user authentication information used for internal systems in centralized servers. Therefore, UXP Portal 2.0 will support authentication and

user management by supporting importing existing user credentials. Portal 2.0 will access user information from the user directory over Lightweight Directory Access Protocol (LDAP). Allowing reusing of an already configured user management directory will save time spent on user management by eliminating the need to manually transmit the changes from the primary user directory to UXP Portal.

2. *TLS certificate management*

   When UXP Portal is offered as a service, mutually authenticated TLS channels are used to secure the connections. This requires means to manage the TLS keys and certificates.

   In UXP Portal 1.0, certification management was possible by connecting to the UXP Portal machine via CLI. As configuring software via CLI is error prone and UXP Portal should be easily manageable by less technical people, the new version will support managing certificates from the graphical user interface (GUI).

3. *UXP Portal customization per organization*

   Organizations are interested in customizing the UXP Portal GUI with their organization specific design elements to maintain branding unity within the organizations information systems. Therefore it will be possible to add organization's logo to the GUI. In addition to that, as UXP Portal is an internationally targeted product, it will support language customization.

4. *User-friendly service user interface (UI)*

   Version 1.0 of UXP Portal displays the service response to the user in a tree structure generated from the XML formatted response message. This presentation of information takes up a lot of space, complicates the search for particular information and does not support filtering. Impractical presentation of information complicates the use of UXP Portal especially for employees of organizations which need to process a large amount of requests every day. Therefore, UXP Portal 2.0 will analyze the XML representations of the request form and response to display an optimal view to the user only based on the information in the WSDL file and received response. For instance, sequences of same XML elements will be displayed as a table to offer

a quick overview of received information. Additionally, fields that require a date can be implemented as date pickers to prevent entering incorrectly formatted date values.

5. *Customization of service UI*

   Based on observations by UXP developers, the average service description in WSDL does not provide enough information to automatically generate optimal service UIs. Thus, Service Administrators will be able to modify the input form and output presentation of the service to support ease-of-use to the Service User. For example, administrators will be able to rename element names and reorder the fields as the names and placement of elements in WSDL file might not match the business logic for the end users. Renaming element names is useful in situations where a service is described in a language different from the organization's internal language.

6. *Responsive design*

   Both using the UXP Portal for administrative activities and consuming the UXP services should be convenient while accessing the application via a web browser of mobile devices with various display sizes. This would be beneficial for the people who spend most of their working day outside of the offices, for instance people occupied in law enforcement, border services or environmental protection agencies.

7. *Audit and request logs*

   As the secure TLS channels used in UXP guarantee privacy of information, organizations requesting sensitive information via UXP Portal might have strong security policies. To meet the security policies, the records of administration related activities will be present in audit logs that are accessible to the administrative roles and the Auditor role. In addition to that, each service request-response pair will be documented by a request log entry visible only to the Service User. The request log will serve as proof of requests made under the name of the user and inform them of suspicious activity.

8. *Auditor role*

The security policies of organizations might require the means for performing special auditing operations. To satisfy this requirement, UXP 2.0 adds the Auditor role. The Auditor can view the audit log and has access to information such as the list of users, list of registered services and access rights. Similarly to Service Administrator and User Administrator roles, Auditor has privileges within one organization.

9. *Exporting signed message containers*

   To prove that data was exchanged, signed and timestamped messages are saved in security server logs. Service Users will be able to download signed message containers and save them for future verification. Service User can either choose a message container that corresponds to their service request or response or both.

10. *Signed message container verification*

    Users might wish to verify the signature of message containers provided by third parties to check validity of their transactions done via UXP. All users will be able to upload a signed message container to UXP Portal which performs the verification and displays the result to the user.

11. *Licensing*

    When customers purchase UXP Portal, they will sign an agreement on terms and conditions of using the software. The agreement will state the organizations who may use the software, the maximum number of organizations and the maximum number of users that can be registered per installation. To avoid violating the agreement, UXP Portal will function only when a valid license file is saved in the system configuration. The system will check the restrictions stated in the license file before permitting operations that might contradict with the license.

2.4 Alternative Tools for SOAP Clients Generation

This section will give an overview of alternative tools that can be used for the core functionality of UXP Portal - dynamically building web service clients based on descriptions written in WSDL. It will cover the functionalities the alternatives

have compared to the lists of initial features in Section 2.2 and new features in Section 2.3.3. It is important to note the requirement that UXP Portal should be easily manageable by people with little technical knowledge. Considering the fact that UXP services are based on SOAP, the focus is on tools that support generating clients for SOAP services. The alternatives can be divided into three categories.

Web services development tools

These are designed to provide software developers the means to quickly implement web services and web service clients. They include methods to send service calls to the service endpoints and to automatically parse response messages.

An example of such is Apache CXF framework [13] which is an implementation of JAX-WS – Java's official application programming interface (API) for XML web services [14]. To simplify building a client application for a specific service, Apache CXF has a CLI tool *wsdl2java* [15] that takes a WSDL file as an input and generates required Java classes.

Another example in this category is Savon [16] – a SOAP client for Ruby that provides methods from setting up the client for a specific service described in WSDL to parsing the received response.

As using frameworks require programming skills and at least basics of implementing web services, they do not satisfy the UXP Portal requirement which aims to eliminate any steps of software development. In addition, using web services is only one part of the capabilities and other functionalities of UXP Portal such as user and service management, TLS configuration would have to be implemented separately.

Web services testing tools

SOAP services testing tools take a WSDL file as an input and display the request in XML format. The testing tools are available as desktop tools and even as extensions for web browsers.

SoapUI is an open source desktop tool that supports testing of SOAP based services [17]. The tool generates an editable XML request form based on the WSDL and provides the means to specify the endpoint of the service. Priced version of the tool generates graphical input fields instead of the XML code.

An example of a web browser extension is Boomerang [18] for Google Chrome. The extension can be used to send service requests similarly to SoapUI. Boomerang lacks features to authenticate server and client by using TLS certificates.

The test applications are suitable for service developers covering the need for testing their services but lack the user management, service access management and in most cases the input form in GUI format required for production use.

HTML form generators

This section covers tools that build a SOAP client along with a corresponding HTML form.

One such web based tool is Generic SOAP Client by SOAPClient [19]. Users can enter service parameters and invoke the request by using their web browser. As a result, the response is shown in XML or HTML format. Lacking any other functionalities, according to the creators, the Generic SOAP Client rather serves as a demonstration how a SOAP client library can be used to implement clients.

Xydra is a library that generates XHTML forms based on a WSDL document [20] and can be used to send a request. Xydra is not in active development, last update dating back to 2003, and therefore they serve at best as an example to developers.

Even if HTML form generators would help with interface generation, multiple other features would have to be implemented, therefore they do not serve as a complete solution for organizations.

Summary of Alternative Tools

In conclusion, the alternatives in all three categories lack a majority of the features that will be supported out of the box by UXP Portal. After the comparison, it became apparent that the common functionality – generating SOAP clients is almost the only common functionality between the UXP Portal and its alternatives. Using alternatives requires programming skills to either implement the main functionality (creating service calls) or the missing functionalities (access rights management, importing available services and others). However, web service testing tools cover the testing usage scenario for service developers and would still be recommended tools for developers who do not have access to UXP Portal. The web services frameworks are useful for developers who have to implement individual service

clients for organizations. Cybernetica organizes trainings and provides tutorial materials on how to use the frameworks and libraries to develop clients. This process will most likely continue as one of the UXP Portal main usage scenarios suggests that UXP Portal might be used as a replacement for final service client applications while they are under development.

3 UXP Portal Development Process

The development process of Cybernetica is based on the Rational Unified Process (RUP) developed originally by Rational Software [21] and currently offered by IBM [22]. The UXP Portal development project uses the same process.

3.1 The Rational Unified Process

According to P. Kroll and P. Kruchten [21, 23], the RUP is an iterative software development approach that offers a wide mixture of different processes that can be accommodated and combined to suit the needs of various software projects.

The structure of the RUP can be illustrated by a two-dimensional diagram where the horizontal axis represents the life cycle of the process and the vertical axis shows logically grouped core processes (see Figure 4).

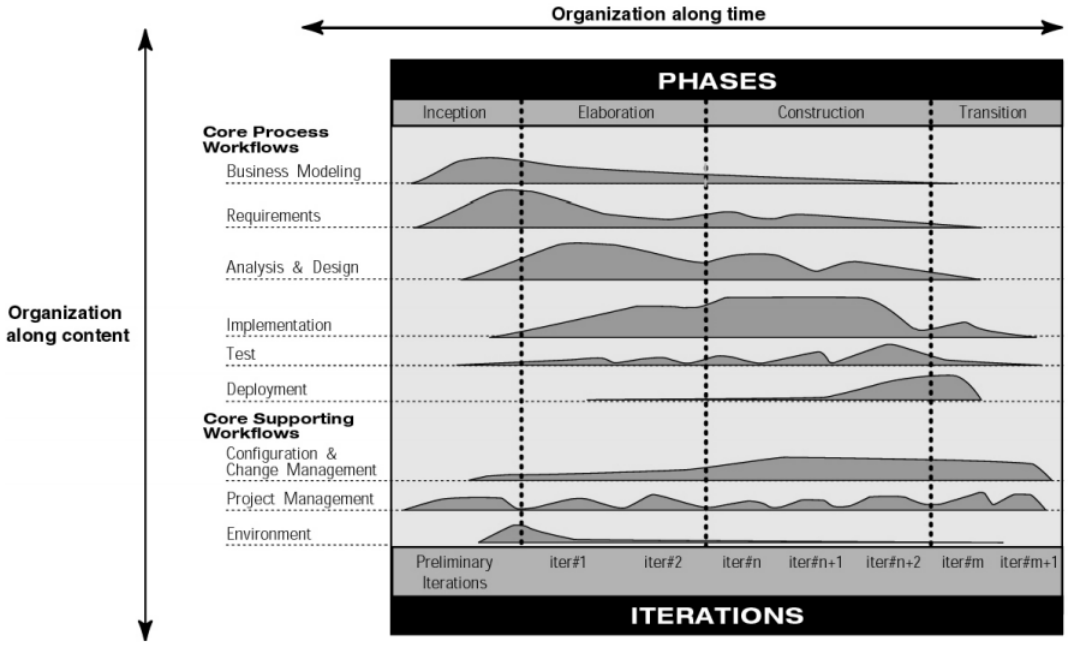

Figure 4. Two-dimensional structure of RUP [2]

The life cycle of a project following the RUP consists of four phases which are in turn divided into one or more iterations. Each iteration ends with a milestone where the progress of the project is measured. The four phases of the project and their main objectives are the following.

1. *Inception*

   Understanding core requirements and the scope of the system and performing business analysis.

2. *Elaboration*

   Detailing requirements and building the first functional prototype of the system.

3. *Construction*

   Building the first operational version of the system.

4. *Transition*

   Creating a final version of the product and deploying it to the users.

The RUP divides project responsibilities between team members by describing a list of *activities* that produce results – *artifacts*. The responsibilities for activities are assigned to *roles* which are divided between team members. An example of a role, artifact and activity combination would be a Project Manager producing a Software Development Plan during Project Initiation activity. To specify the sequence of activities, they are gathered into workflows. The main workflows of the RUP, presented on the vertical axis of Figure 4, are the following:

1. Business Modeling
2. Requirements
3. Analysis & Design
4. Implementation
5. Test
6. Deployment
7. Project Management
8. Configuration and Change Management
9. Environment

The Initiate Project activity introduced in the last example is a part of the Project Management workflow. Activities performed in the scope of this Thesis are part of the Business Modeling and Requirements core workflows which according to the RUP are main workflows for an analyst role.

3.2 Project Artifacts and Activities

So far the UXP Portal project has passed the Inception phase and is currently in the Elaboration phase. Figure 5 gives an overview of different artifacts created within the first two phases of the project and their dependencies. The solid arrow marks that the artifact serves as an input for another artifact. The artifacts in blue were created by the author of this Thesis.

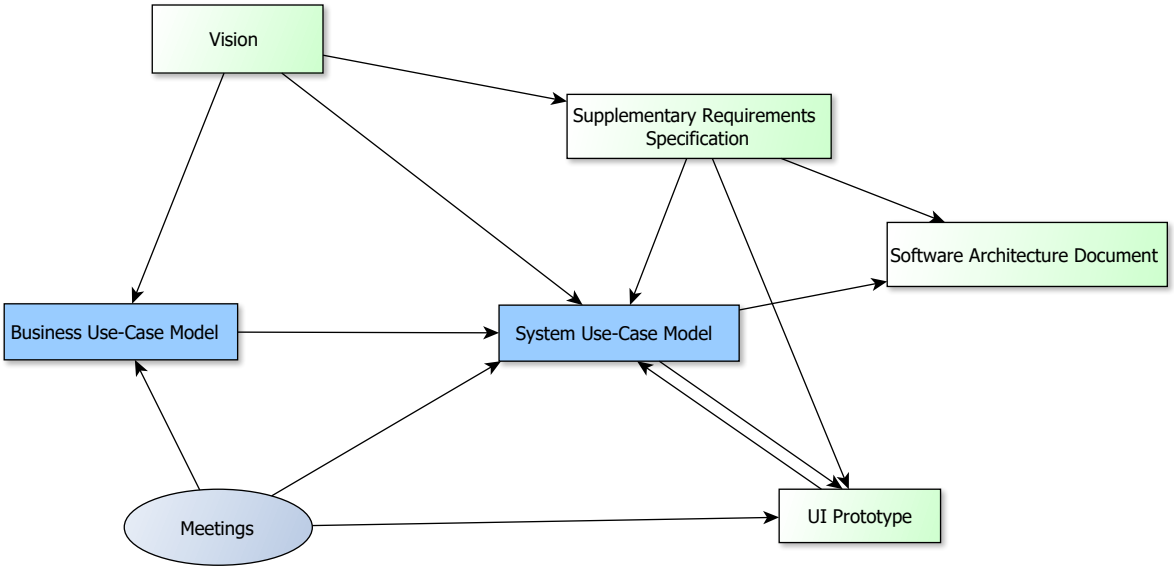

Figure 5. Relationships between the project artifacts created in the Inception and the Elaboration phases

Vision During the Inception phase, a Vision document was produced by the project manager with high-level usage scenarios and a feature list for the product. This document served as input for the Business Use-Case and System Use-Case Models. The most important features of the new version of UXP Portal are listed in section 2.3.3. The development of subsequent artifacts must take into consideration feature priorities described in the Vision.

Business Use-Case Model Based on the Vision, the author of this Thesis constructed a Business Use-Case Model to outline the business processes related to the UXP Portal. The model includes a Business Domain Model, which describes the business workers and objects related to them, and descriptions of business use cases. As the UXP Portal will be designed to support two different deployment scenarios described in Section 2.3.2, both scenarios and their differences were detailed. The Business Use-Case Model is useful for explaining to the customers how the UXP Portal is used in the UXP infrastructure.

System Use-Case Model The functional requirements of the product under development are described by a System Use-Case Model created by the author of this Thesis. The model includes a System Domain Model which gives an overview of objects relevant to system use cases, including relationships between them and their attributes, and detailed descriptions of system use cases. The use cases were developed to cover the feature list in the Vision. The System Use-Case Model will be delivered as a description of functionalities for the developers implementing the UXP Portal. The priorities of the features in the Vision are present in the use case descriptions and serve as guidelines for developers.

Meetings As UXP Portal 2.0 is being developed as a product with no direct customer participation, eliciting requirements had to be done by interviewing team members with a wide range of experience in the UXP domain. This included meetings with people who have been giving trainings to different UXP end user roles and people who have been actively involved in the development of UXP Portal 1.0 and other UXP products. Their experience includes providing technical support during the installation and initialization processes of the UXP installations. These team members' diverse understanding of UXP domain makes meetings with them the most valuable source for requirements.

Supplementary Requirements Specification A Supplementary Requirements Specification is managed by the architect of the project. The document includes different technical and non-functional requirements for the system. The requirements in this document act both as an extension and input for

the Use-Case Model and the Software Architecture Document.

Software Architecture Document The Software Architecture Document is maintained by the architect of the project and provides different views of the system, including logical view, implementation view and deployment view. The views must take into consideration specifications in the System Use-Case Model and Supplementary Specification.

UI prototype The UI prototype was created in collaboration with a company specialized on user experience (UX) and UI design. This included weekly meetings between the interaction designer, the author of the Thesis as an analyst and the project architect. Feedback was provided both during the meetings and as written feedback on the updated prototype. Since according to the RUP, developing UI prototypes is done in parallel with detailing the use cases [21], both the prototype and use case model had an impact on each other.

3.3 Modeling

The RUP encourages visualizing and documenting software specifications with models using UML [23]. Therefore, to produce artifacts, the author has used various UML diagrams. Descriptions of business use cases are supported by *activity diagrams*. *Class diagrams* are used to illustrate relationships between domain objects in both Business and System Use-Case Models. Actors and their use cases are depicted as *use case diagrams*. The guidelines for using UML were obtained from Martin Fowler's book *"UML Distilled"* [24] which is also translated to Estonian by Cybernetica and used as a recommended reference within the company.

4 Use-Case Models Overview

This section gives an overview of the Business Use-Case and System Use-Case Models developed to specify the functional requirements of UXP Portal 2.0. As the full documents are about 70 pages long when combined, the detailed descriptions are omitted and replaced with shorter explanations. For both domains, a model is introduced to clarify concepts relevant to the use cases. The use cases are listed as use case diagrams. In the business domain, the focus of describing the use cases is to differ between the two deployment scenarios of UXP Portal. System domain use cases are grouped by the different roles interacting with the system-to-be. A sample use case description with a complementary activity diagram is added for both domains to give an insight of the used technique.

4.1 Business Use-Case Model

4.1.1 Business Domain Model

The class diagram in Figure 6 shows concepts relevant to the Business Use-Case Model. This includes business actors, business objects and relationships between them.

UXP Portal Business Use-Case Model includes the following business actors:

UXP Member Natural or legal person that has entered into a UXP membership agreement with the Governing Authority to use UXP functionalities or a natural or legal person that wishes to enter into that agreement. Can be a Portal Client, Portal Service Provider, or a UXP Service Provider.

Portal Client Natural or legal person who has signed a subscriber agreement to use Portal software provided by Portal Service Provider or a natural or legal person who wishes to establish that subscription.

Portal Service Provider Organization that provides UXP Portal software as a service or an organization that wishes to start providing that service.

Portal Owner Natural or legal person who has configured or wishes to configure the Portal for their own use. In that case the Portal Owner fills the roles of both the Portal Client and the Portal Service Provider.

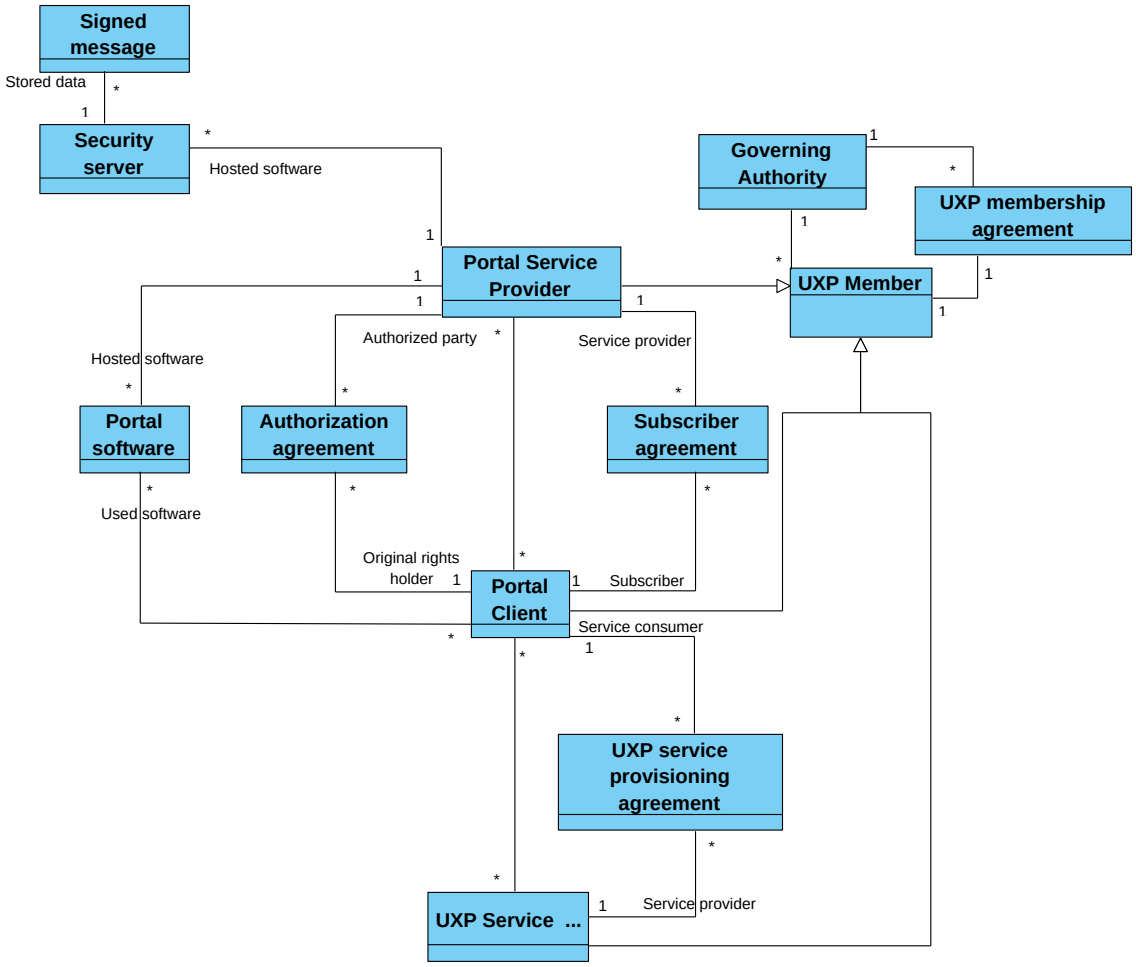

Figure 6. Business domain model

UXP Service Provider UXP member that provides services over UXP infrastructure.

Governing Authority Institution that manages the UXP instance and registers UXP members.

The objects related to business actors are the following:

Portal software Instance of UXP Portal software used by the Portal Client according to the subscriber agreement. A UXP component for accessing UXP services. The system under design.

Security server UXP component for routing UXP messages. The Portal Service Provider registers Portal Clients in the security servers to connect them to the UXP infrastructure.

Signed message container A timestamped message signed with the message originator's signing key. Signed message containers are stored in security servers and can be downloaded.

UXP membership agreement An agreement concluded between the UXP Member and the Governing Authority that specifies the terms and conditions of using UXP infrastructure.

UXP service provisioning agreement An agreement that specifies the legal relationship between the UXP Service Provider and the Portal Client.

Subscriber agreement A software subscription agreement concluded between the Portal Client and the Portal Service Provider.

Authorization agreement An agreement concluded between the Portal Client and the Portal Service Provider where the Portal Client authorizes the Portal Service Provider to request a signing certificate on behalf of itself.

4.1.2 Business Use Cases

Since UXP Portal 2.0 must support two deployment scenarios introduced in Section 2.3.2, business scenarios have been modeled by describing two different use case diagrams.

SaaS installation

Figure 7 contains use cases required for the scenario where the organizations providing UXP Portal as a service and the client of the UXP Portal are two separate entities.

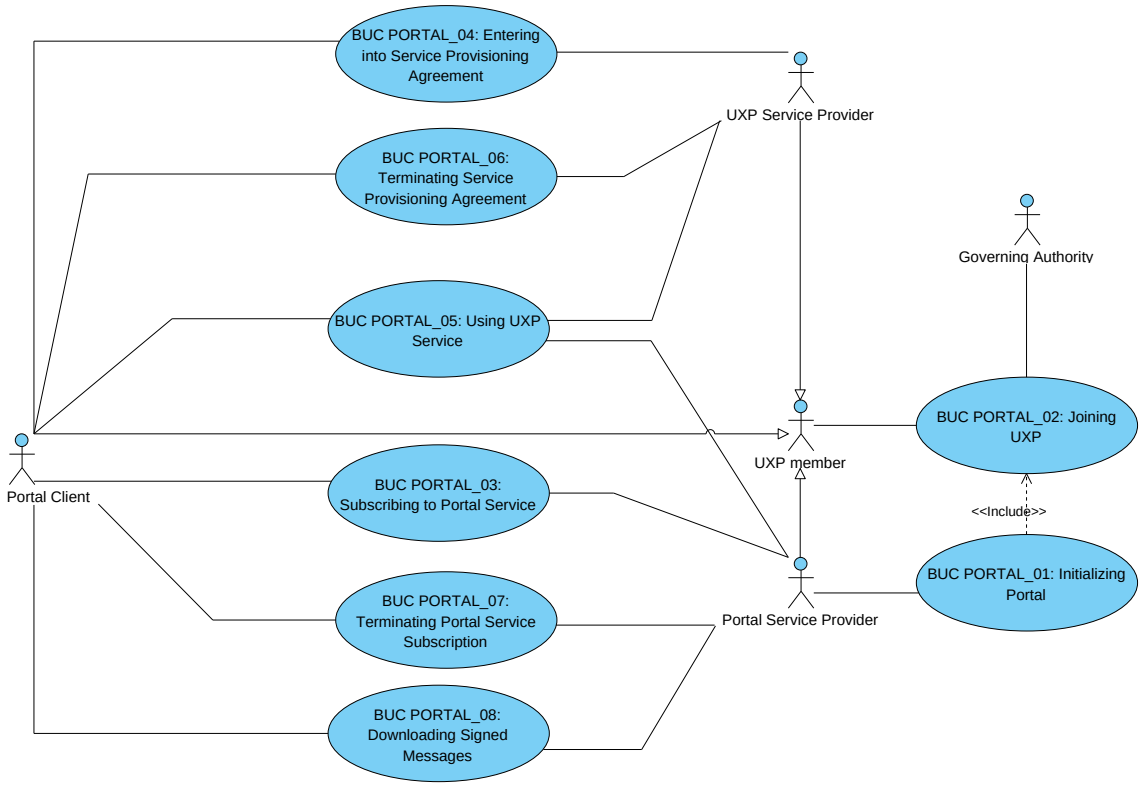

Figure 7. Business use case diagram: UXP Portal as SaaS

BUC PORTAL_01 describes the processes required to start providing UXP Portal as a service. This includes installing and configuring the software and joining the UXP infrastructure if the Portal Service Provider is not yet a member. In addition, providing the UXP Portal service requires installing a security server and registering it to the UXP infrastructure.

BUC PORTAL_02 describes the process of becoming a UXP member which is a requirement for every actor presented in this use case diagram. UXP membership is not a requirement for installing UXP software but it is a requirement for operating it.

BUC PORTAL_03 and BUC PORTAL_07 describe initiating and terminating the agreement between the Portal Client and Portal Service Provider on the terms and conditions of using the UXP Portal software. The process of subscribing to the Portal service includes steps like informing the GA about a new Portal Client, registering the Portal Client in a security server and UXP Portal and granting access to the administrative personnel of the Service Client.

BUC PORTAL_04 and BUC PORTAL_06 describe initiating and terminating the agreement between the Portal Client as a service client and UXP Service Provider as a service provider. These are standard procedures done by two UXP members who wish to exchange information over the UXP whether the consumer side uses UXP Portal or not.

BUC PORTAL_05 describes the main functionality of UXP Portal software – using a UXP service via UXP Portal, on a business level.

BUC PORTAL_08 describes the process of downloading signed message containers.

Local installation

In Figure 8, the Portal Service Provider and Portal Client are merged into one business actor – Portal Owner. The following changes were applied to the use cases in the SaaS installation model.

BUC PORTAL_03 BUC PORTAL_07 are removed as processes for subscribing to UXP Portal service and terminating from the service become redundant in this scenario.

BUC PORTAL_03a describes a process of starting to use UXP Portal without the intermediate Portal Service Provider. Processes of informing the GA and registering in a security server are done by the Portal Owner. Subsystem is an entity that corresponds to an organization in UXP Portal.

BUC_PORTAL_05a is a modification of BUC PORTAL_05. Portal Service Provider is removed as an intermediate actor and the messages are exchanged directly between the service provisioning agreement partners.

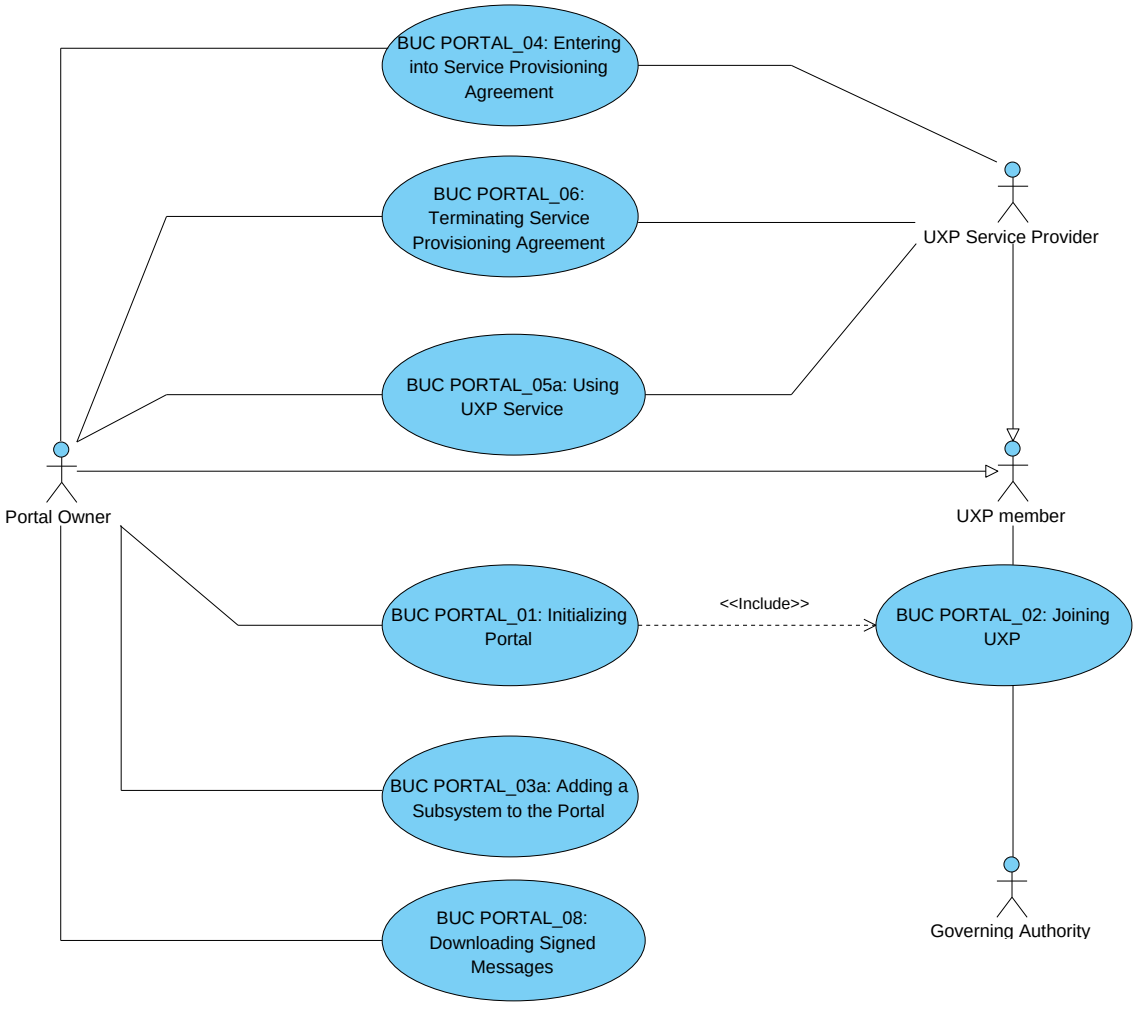

Figure 8. Business use case diagram: UXP Portal as local installation

4.1.3 Sample Business Use Case

In the Business Use-Case Model document, each of the use cases are described in detail and illustrated with an activity diagram. The following is an example of a business use case description, detailing the process how an organization subscribes to UXP Portal service.

Business Use Case: Subscribing to Portal Service

Actors Portal Client, Portal Service Provider

Main Success Scenario

1. Portal Client subscribes to the UXP Portal service and authorizes Portal Service Provider to request a signing certificate on behalf of itself.
2. Portal Client informs Governing Authority about subscribing to UXP Portal service.
3. Portal Service Provider requests a certificate from the certification service provider using the authorization from Portal Client.
4. Portal Service Provider registers Portal Client in a security server.
5. Portal Service Provider registers Portal Client in the UXP Portal.
6. Portal Service Provider creates User Administrator and Service Administrator accounts for Portal Client and sends the credentials to Portal Client.

Extension

6a. Portal Client wishes to use existing LDAP directory for user management.

- 6a1. Portal Service Provider adds connection between a LDAP directory and the Portal to configure user management.
- 6a2. Use case terminates.

The corresponding activity diagram is in Figure 9. The first black rectangle is a *fork* that indicates that the following activities can be done in parallel [24]. The process flows are synchronized by a next rectangle denoting a *join*.

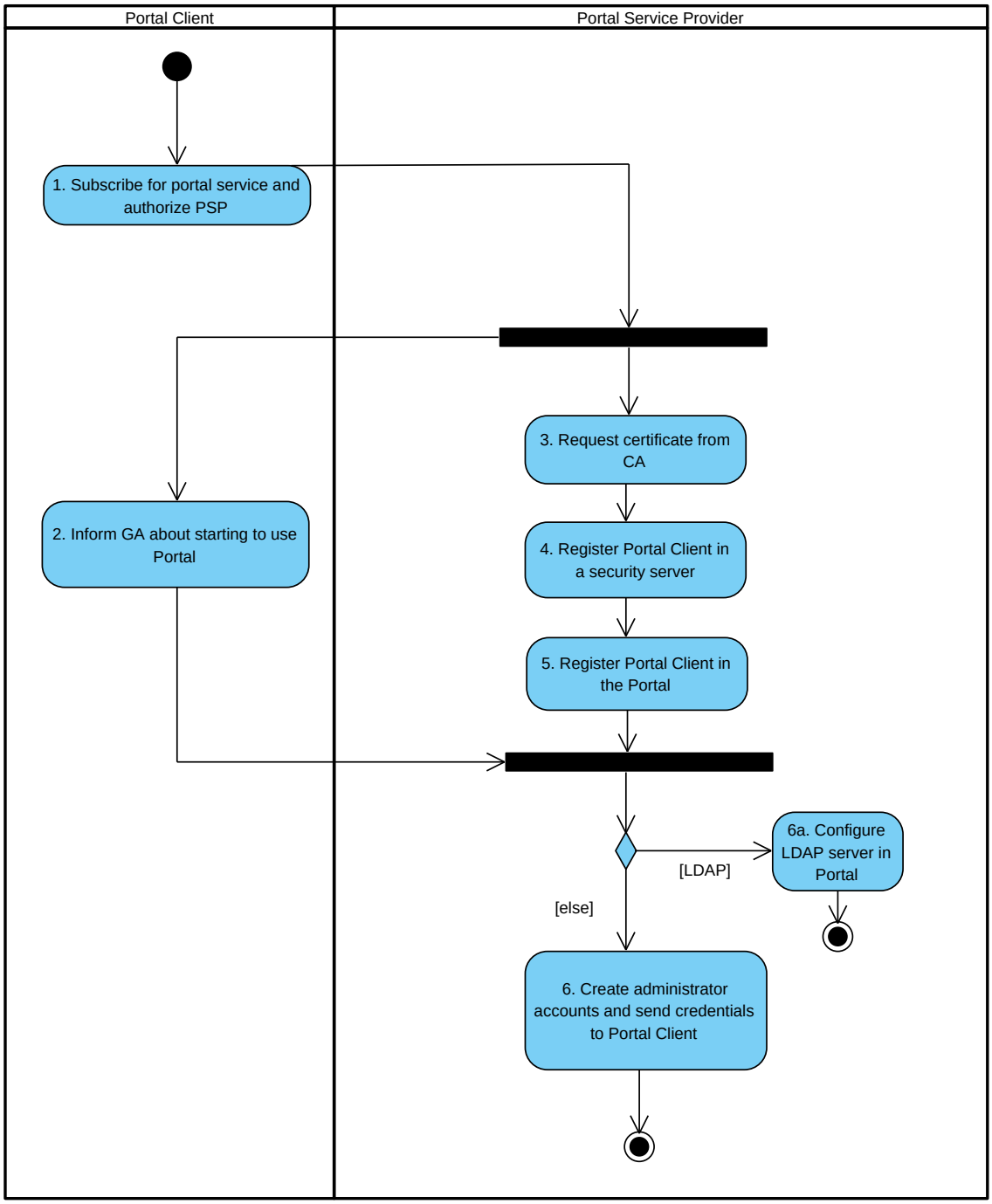

Figure 9. Activity diagram of the business use case *Subscribing to Portal Service*

4.2 System Use-Case Model

4.2.1 System Domain Model

The conceptual class diagram in Figure 10 shows the relationships between the UXP Portal 2.0 system domain entities and their attributes present in the system use cases.

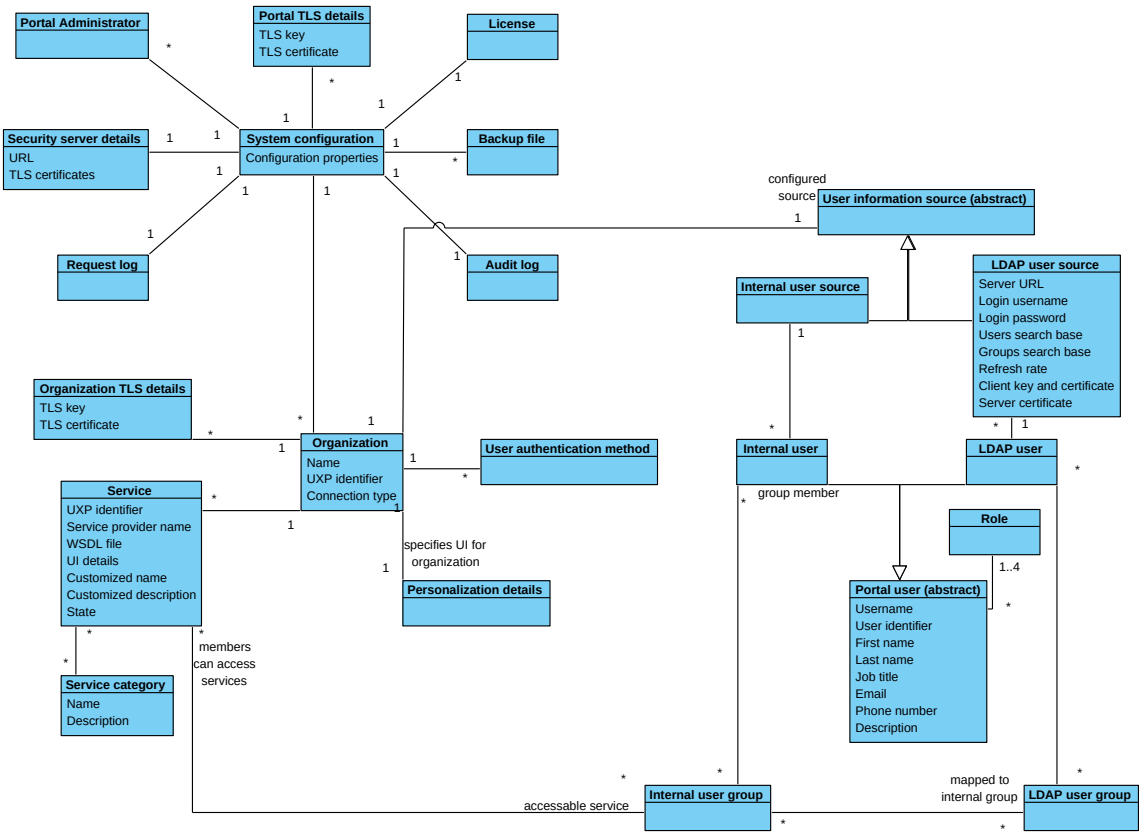

Figure 10. System domain model

System configuration Contains values to system properties specific to the UXP Portal instance. General collection of information stored in UXP Portal.

License file Required for functioning of UXP Portal. License can set boundaries to the use of UXP Portal software. Without a valid license file, UXP Portal supports only the functionalities necessary for license file upload.

Security server details Information required to establish a connection between UXP Portal and a security server.

URL — URL used to establish connection to the security server.

TLS certificates — security server TLS certificate is received from the person responsible for security server. UXP Portal uses one security server certificate at a time to establish connections on behalf of all organizations.

Portal TLS details Part of the UXP Portal configuration necessary for establishing secure connection between the UXP Portal and the web browser that is used to access the UXP Portal GUI.

TLS key — generated in UXP Portal or imported.

TLS certificate — either self-signed by UXP Portal or imported TLS certificate.

Organization Entity in UXP Portal instance that corresponds to a UXP member's subsystem.

Name — name of the organization in Portal, for example the official name of the company or the name imported from the security server during the registration of the organization.

UXP identifier — identifier of the subsystem that represents the organization in UXP instance.

Connection type — connection type between UXP Portal and the security server. Possible values are HTTP, HTTPS or HTTPS NO AUTH. Must match the connection type set for this organization in security server.

Organization TLS details UXP Portal TLS details for establishing HTTPS connection for the organization between the UXP Portal instance and the security server. Each organization can have multiple key-certificate pairs stored in UXP Portal.

TLS key — generated in UXP Portal or imported.

TLS certificate — self-signed by UXP Portal or imported.

User information source UXP Portal supports two different user information sources for organizations: internal user source and LDAP user source.

Internal user source User source in case users will be created in UXP Portal GUI and their information and authentication details are stored in the UXP Portal internal database.

LDAP user source User source in case user information and authentication details are periodically imported from external LDAP directory. When external directory is used, user information can not be modified in UXP Portal.

Server URL — URL used to connect to the directory server including the port number.

Server certificate — server certificate uploaded to UXP Portal when establishing server requires TLS authentication.

Client key and certificate — TLS certificate that will be used by UXP Portal to authenticate itself as a LDAP server client on behalf of the organization the user source is configured for.

Login username — distinguished name of the user that will be used to establish the connection to the directory server.

Login password — password of the aforementioned user.

Users search base — location in the directory server from where the search for users starts.

Groups search base — location in the directory server from where the search for user groups starts.

Refresh rate — time interval after which the user and user group information is updated in UXP Portal.

User authentication method Login method configured for the users. The default authentication methods are:

in case of internal user source — username and password stored in UXP Portal database;

in case of LDAP user source — username and password stored in LDAP server.

Portal user Abstract user class which realizations are *internal user* and *LDAP user*.

Username — log in name that is imported from LDAP or created in UXP Portal, must be unique within one organization.

User identifier — unique user ID for SOAP message header.

First name

Last name

Job title — user's position in the organization.

E-Mail — communication channel for communication between the service users and administrators.

Phone number — alternative means of communication.

Description — general field for organization specific information or comments about the user.

Role Permission to access a specific set of UXP Portal functionalities. UXP Portal has 5 built-in roles: Portal Administrator, Service Administrator, User Administrator, Service User, and Auditor. Service User role is assigned by default to all users created within an organization. Service Administrator, User Administrator and Auditor roles are assigned by adding the user in the respective administrative user groups. Portal Administrator role is assigned during the creation of Portal Administrator account. All roles except Portal Administrator are assigned within one organization.

Internal user User account created in UXP Portal and stored in UXP Portal internal database.

Internal user group User group created in UXP Portal. Rights to access services are granted to internal user group members by granting the right to the group as a whole. Internal groups are created within organizations, each organization has three built-in groups for Service Administrators, User Administrators, and Auditors.

LDAP user User imported from a LDAP directory server. LDAP users can not be directly associated with user groups in UXP Portal.

LDAP user group Group imported from a LDAP directory server. LDAP groups can be mapped to internal user groups in UXP Portal and thereby the members of these LDAP groups receive the service access rights granted to the internal group.

Personalization details Organization specific user interface attributes. For example the logo of the organization which will be displayed in the user interface for all users.

Service UXP service that is registered in UXP Portal.

UXP identifier — full identifier of the service in UXP.

Service provider name — name of the service provider as specified in the service provider's security server.

WSDL file — description of the service in Web Service Description Language which in addition to service input and output descriptions includes human readable information about the service:

Title

Description

Technical description

Customized name — unique name of the service in UXP Portal. If the user does not specify a customized name for the service, the title imported from WSDL file is used.

Customized description — description of the service in UXP Portal. If the user does not specify a customized description for the service, the description imported from WSDL file is used.

UI details — automatically generated or edited machine processable description of UI for service client.

State — ON/OFF.

Service category Set of services created for visual grouping of services in UI.

Name

Description — longer text field for details.

Backup file File that can be exported from UXP Portal containing information required to restore the system configuration.

Request log Log that contains entries of service requests and responses.

Audit log Log that contains entries of changes in system configuration.

4.2.2 System Use Cases

Access to the functionalities in UXP Portal 2.0 GUI are assigned by five built-in user roles. The roles correspond to the actors performing the system use cases. A Systems Administrator actor is added to group the use cases performed outside of the GUI of UXP Portal, like installation and restoring the system from a backup file. The relationships between the actors are displayed in Figure 11. Functionalities that can be used by all users in GUI are gathered under a separate use case diagram where the primary actor is *User*.

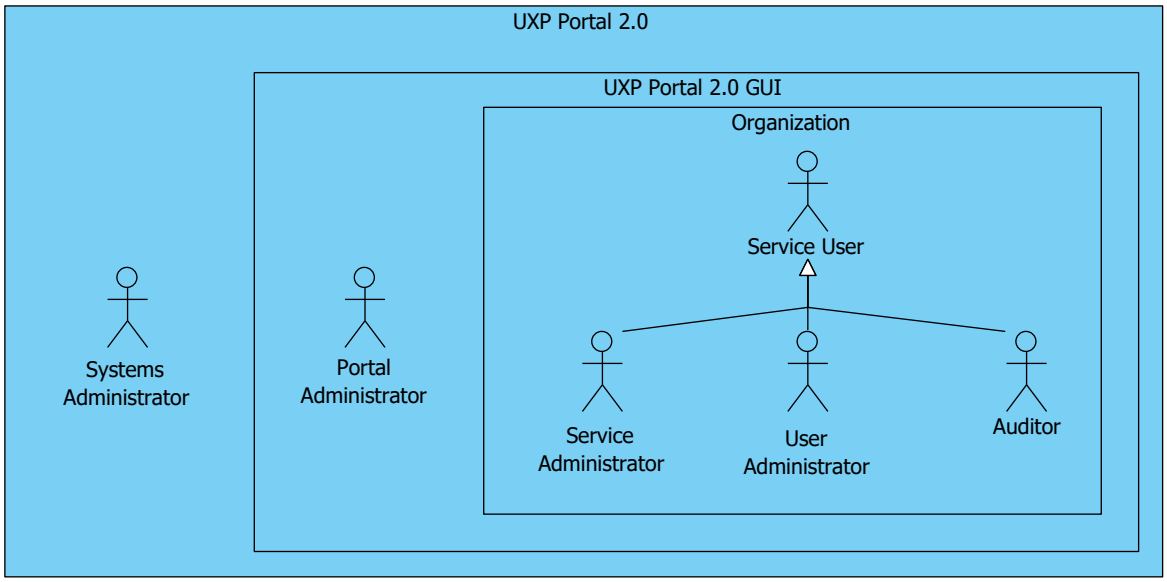

Figure 11. Actors in system use cases

The author has used *include* and *extend* relationships to relate the use cases in the use case diagrams. According to Jason T. Roff, the *include* relationship between use cases indicates that a use case uses the functionalities of the included use case to fulfill its main goal. Gathering repeated functionality in a single use case and including it from multiple other use cases is useful for avoiding repetition.

The *extend* relationship implies that the use case may be lengthened by another use case [25].

Portal Administrator performs the first initialization of UXP Portal which includes uploading a valid license file, configuring the connection to a security server and registering the first organization.

As the Portal Administrator is the first user of the system, he has the privileges to either import existing user accounts from a LDAP server or manually create new user accounts. To pass on the administrative responsibilities within an organization, Portal Administrator assigns User Administrator and Service Administrator roles to authorized users by adding them to designated user groups.

Portal Administrator can edit the UI branding details for an organization and set whether the connections established on behalf of the organization use TLS or not.

Portal Administrator role has control over all organizations registered in a UXP Portal installation. All other user roles can perform functionalities only within the organization they are registered in.

For Portal Administrator use cases, see Figure 12. The actor stick figure is removed from that diagram to improve the readability of the relationship notations.

User Administrator If built-in user information source is used, User Administrator has the functionalities to create, edit and delete user accounts. User Administrator can group users in user groups, for example group all employees of one department under one group. User Administrator has also the privileges to assign User Administrator, Service Administrator and Auditor roles by adding users to the corresponding built-in user groups.

If user information is imported from a centralized server, User Administrator does not have privileges to manipulate user accounts. In that case, users are assigned to internal user groups by mapping LDAP groups to internal user groups in Portal. Since adding a new user to a user group grants the user all the privileges of the user group, User Administrator must have a good

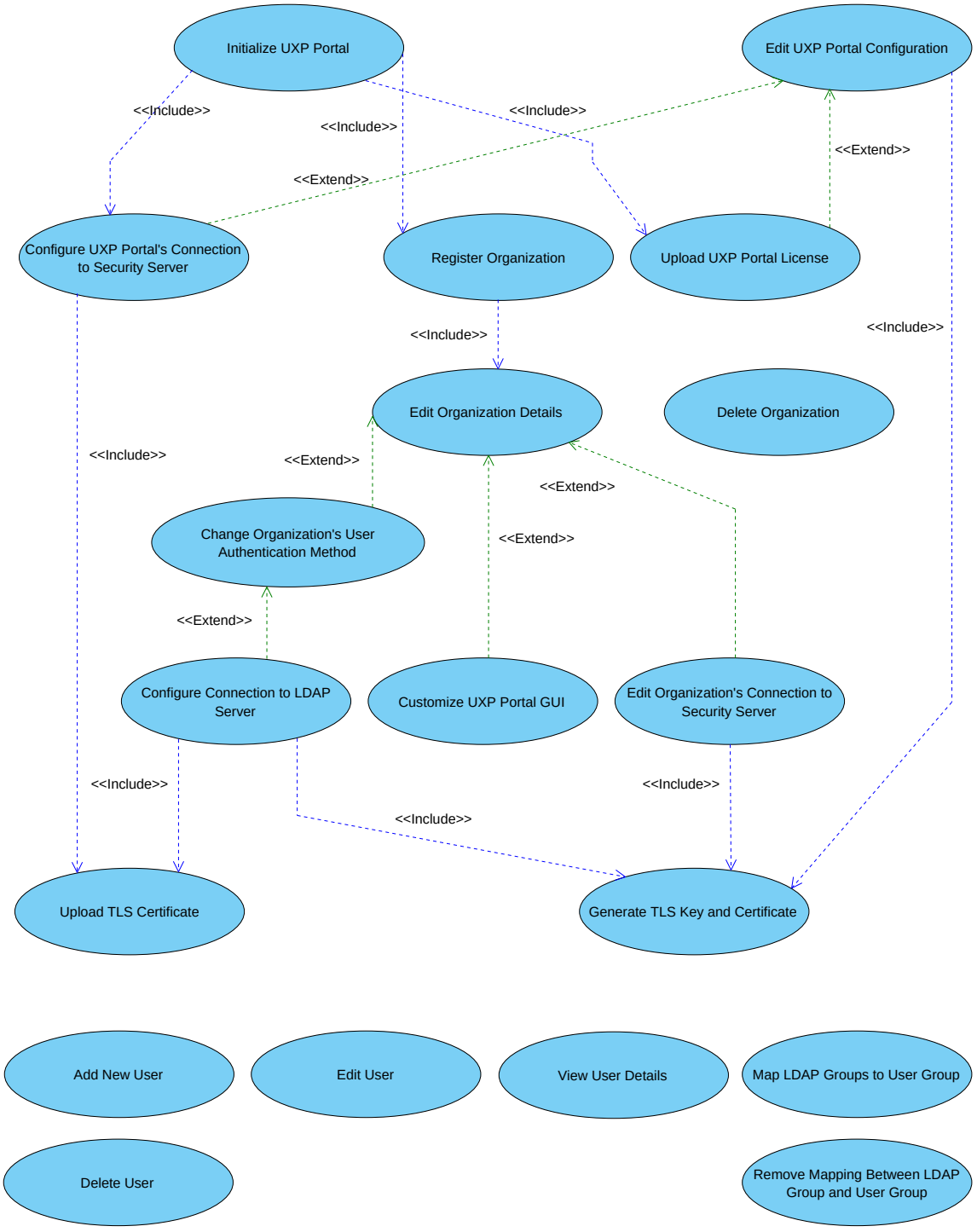

Figure 12. Portal Administrator use cases

knowledge of the employees behind the user accounts and their authorization levels in the organization.

User Administrator's use cases are listed in Figure 13.

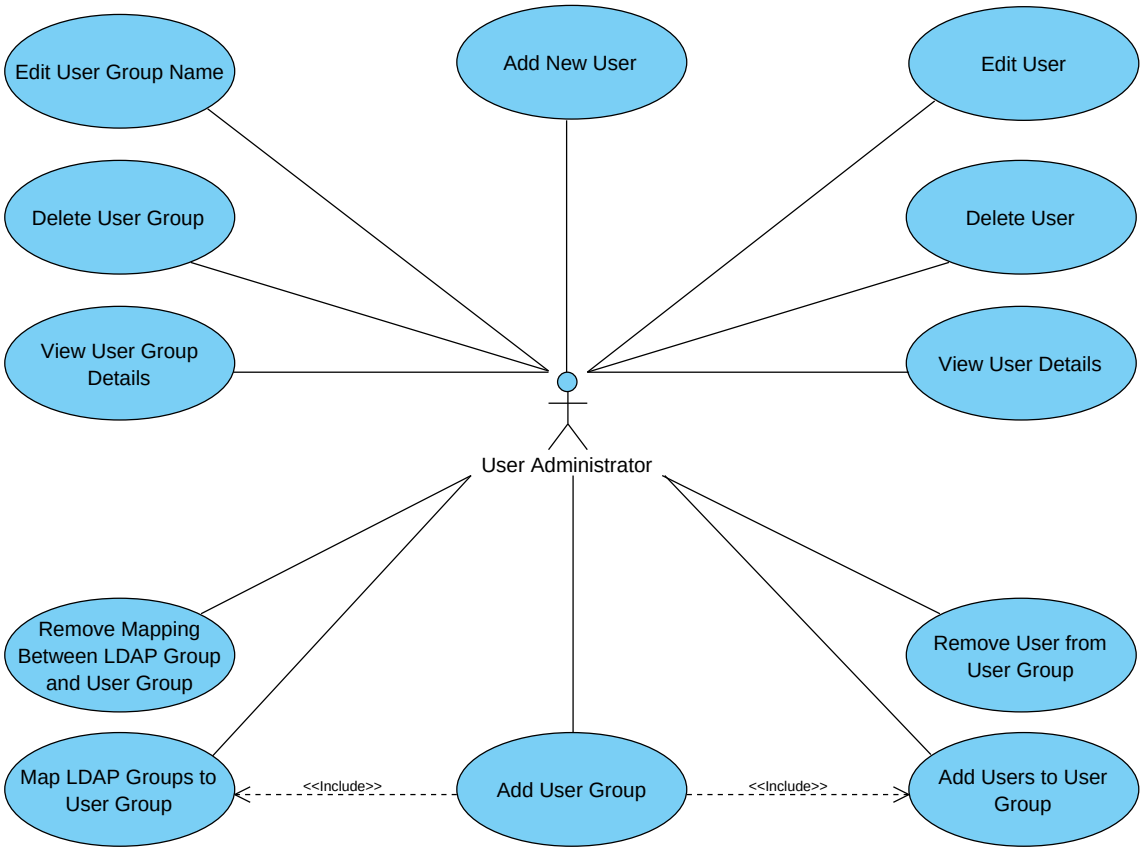

Figure 13. User Administrator use cases

Service Administrator manages information about service providers and services registered in UXP Portal and assigns service's access rights to Service Users. This includes registering new services and customizing their UIs to improve the ease of use for Service Users.

Service Administrator can group registered services in categories. The services are displayed to Service Users by categories to simplify locating services related to certain workflows within the organization.

Service Administrator also has permission to grant access rights. Access rights are only granted to user groups and not to every user separately. Service Administrator has to be familiar with the logic behind the UXP services and receive information about the details agreed upon in the service provisioning agreement between the client organization and service provider. This agreement specifies which Service Users should be granted access to the service.

Service Administrator's use cases are listed in Figure 14.

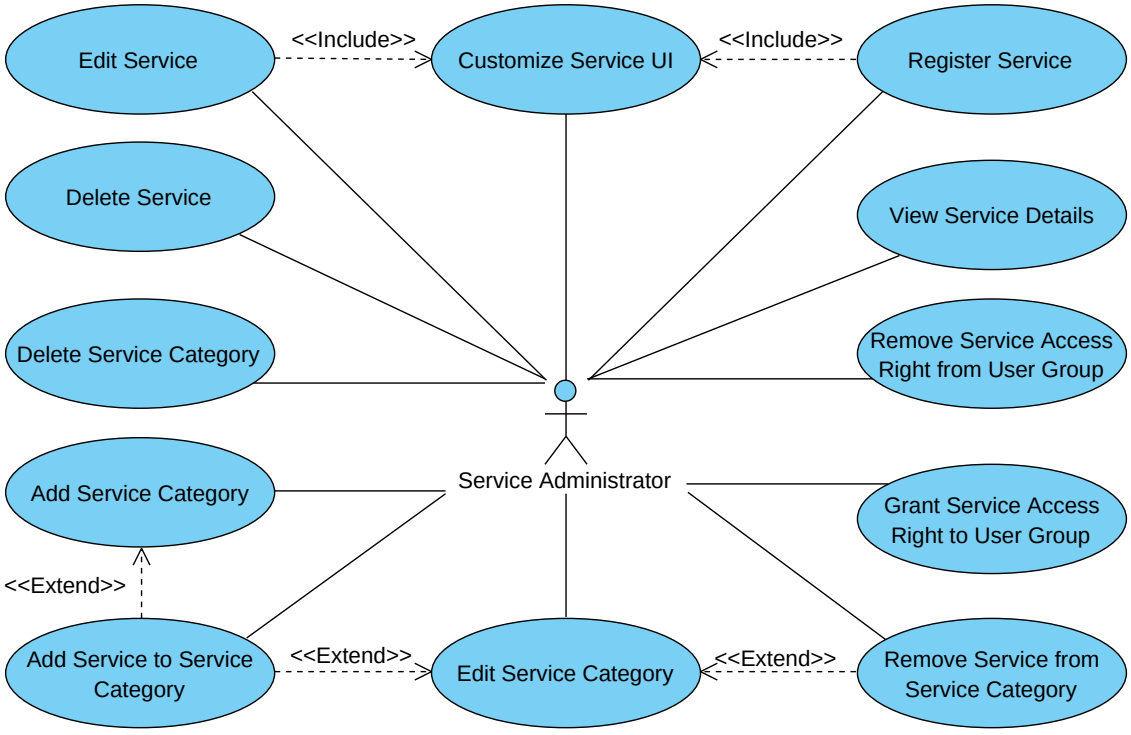

Figure 14. Service Administrator use cases

Service User can access UXP services, that they have been granted access to, via a graphical user interface. Service Users can view their usage of services from the request log and download the signed message container to save it in the local file system.

Auditor can view audit log and other information visible to the administrators like user and service information.

Systems Administrator installs the UXP Portal software and manages backups. All Systems Administrator use cases are performed by using the CLI.

User is the primary actor type for the use cases performed by all actors except the Systems Administrator. Each user can log in and log out of the system, change their password and view their account details stored in UXP Portal or imported from the external server.

The functionality of verifying a signed message container is available to all users. Message container can be downloaded either right after a response was received or later from the request log.

The use cases performed by the Service User, Systems Administrator and User are combined to a use case diagram in Figure 15.

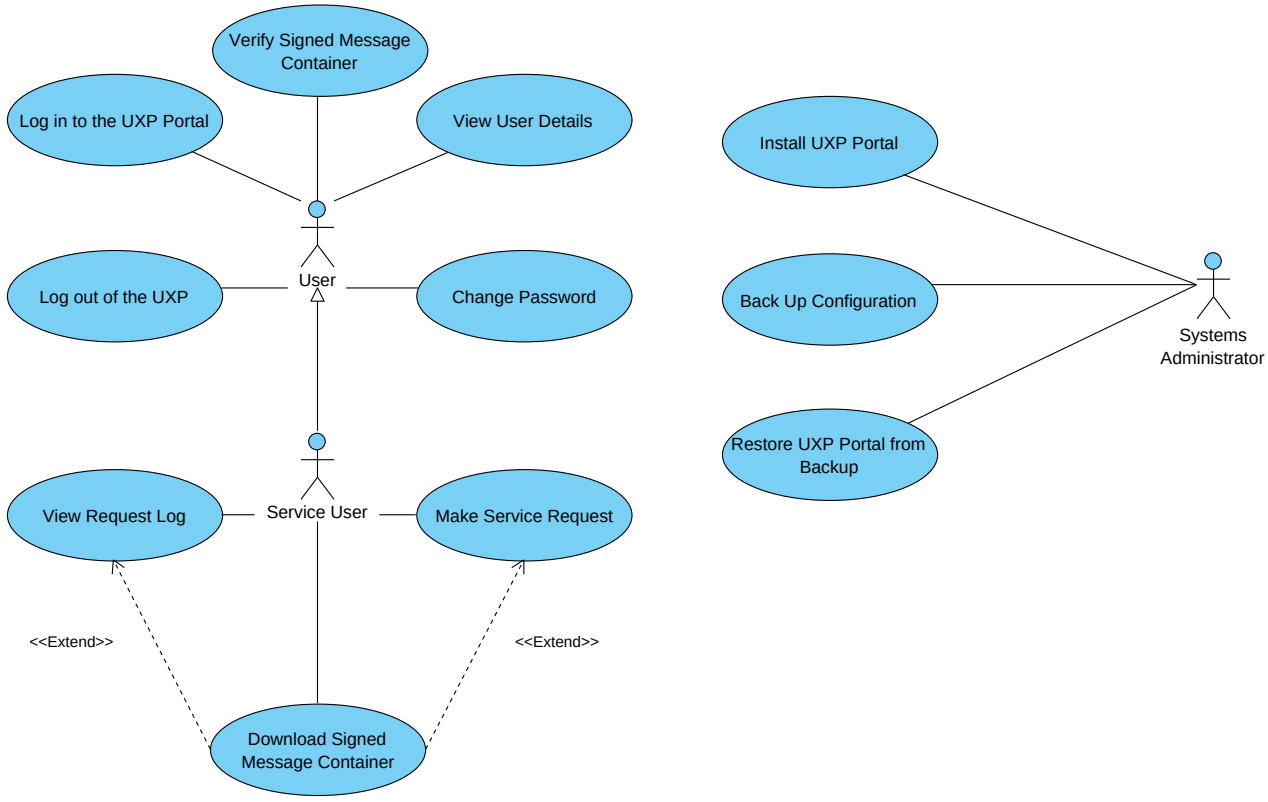

Figure 15. Combined use case diagram for the rest of the actors

The privileges to view lists of different entities stored in UXP Portal are illustrated as a separate use case diagram (see Figure 16). The use case descriptions specify attributes and actions displayed for each entity.

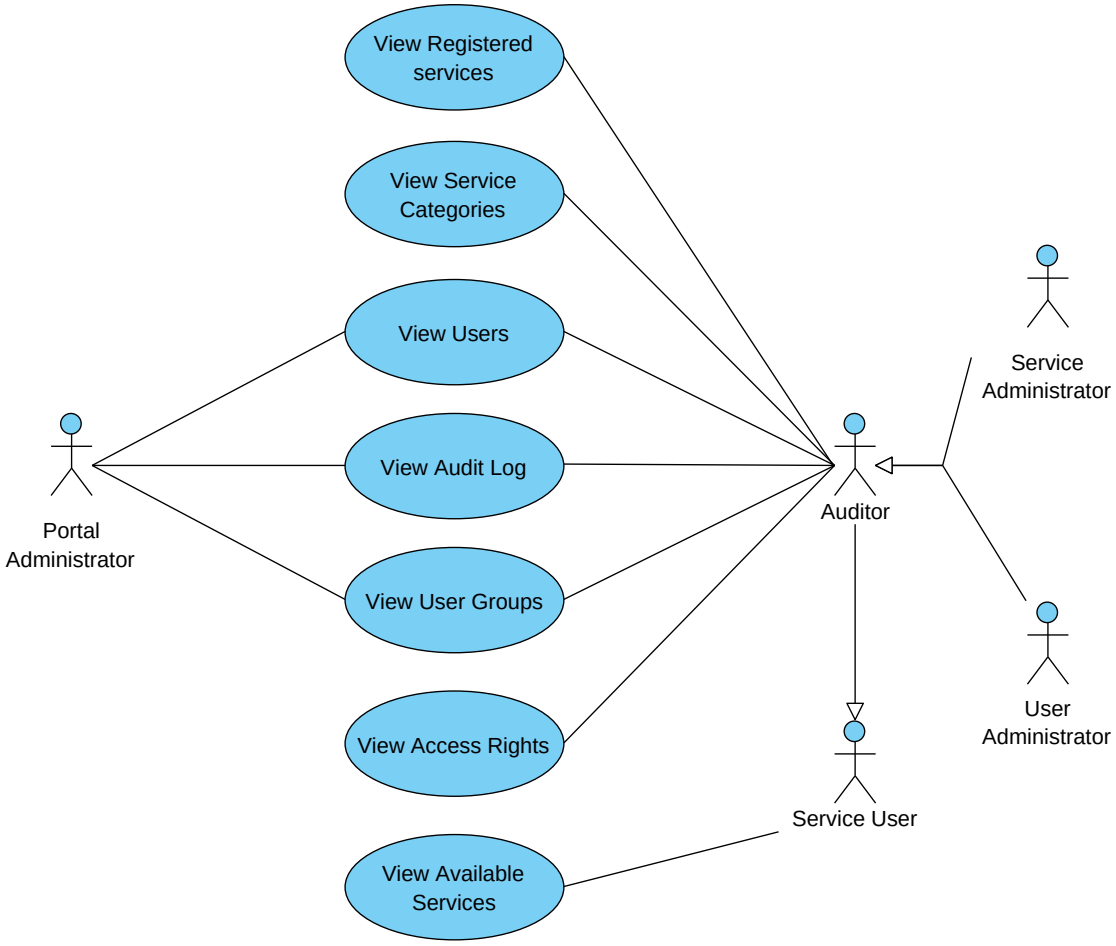

Figure 16. Use cases for viewing lists of entities

4.2.3 Sample Use Case Description

Each of the use cases has a level attribute, chosen between two of the three levels introduced by Cockburn [26]: *user goal* level depicting a functionality for fulfilling user's job responsibilities and *subfunction* level, that was assigned to the use cases included by multiple other use cases. An example of a subfunction level use case is *Upload TLS certificate.* The third level suggested by Cockburn — *summary* level, corresponds to the business use cases described in the previous section.

The assigned priorities are *high* and *medium* to differ between functionalities that have to be implemented for the first fully functional version of the system and the convenience functionalities that will be added in the following iterations.

The *Register Organization* use case is added as an example of a system use case on a *user goal* level. Use case *Register Organization* is the realization of the step 5 of business use case *Subscribing to Portal Service* previously shown in Section 4.1.3.

System Use Case: Register Organization

Level User goal

Priority High

Primary Actor Portal Administrator

Main Success Scenario

1. Portal Administrator selects to register an organization in UXP Portal.
2. System verifies the UXP Portal license file.
3. System forms an autosuggestion list of possible UXP identifiers based on the following information:
    - restrictions specified by the license;
    - list of clients imported from the security server (if security server URL is configured);
    - organizations already registered in the Portal (excluded).
4. Portal Administrator enters organization's UXP identifier components.

5. System suggests possible UXP identifiers based on the autosuggestion list.
6. Portal Administrator selects an organization to be registered in the Portal from the suggestions.
7. System displays the following values about the organization:
   - UXP identifier;
   - name.
8. Portal Administrator edits the name of the organization in UXP Portal.
9. System saves the new organization in the system configuration.
10. System logs the event "Register organization" to the audit log.
11. System redirects the Portal Administrator to the organization's details page.
12. Portal Administrator edits the organization's details (Use Case: *Edit Organization Details*).

Extensions

2a. The maximum number of organizations registered in the Portal is reached.

2a1. System displays the error message.

2a2. Use case terminates.

2b. License file contains restrictions on the UXP identifier of the organizations that can be registered in the Portal.

2b1. System prefills the components of the UXP identifier with the values predefined by the license file.

2b2. Use case continues from step 3.

2c. License file specifies only one organization that can be registered.

2c1. Use case continues from step 7.

6a. Portal Administrator continues specifying the organization manually.

6a1. Portal Administrator enters the following information about the organization:

- UXP identifier;
- name.

6a2. Use case continues from step 9.

8a. Portal Administrator does not wish to edit the name of the organization.

8a1. Use case continues from step 9.

9a. An organization with the entered name already exists in the system configuration.

9a1. System displays the error message.

9a2. Use case continues from step 8.

9b. An organization with the entered UXP identifier already exists in the system configuration.

9b1. System displays the error message.

9b2. Use case continues from step 4.

Figure 17 contains the activity diagram of the *Register Organization* use case. The transitions between the activities of main success scenario are represented in bold.

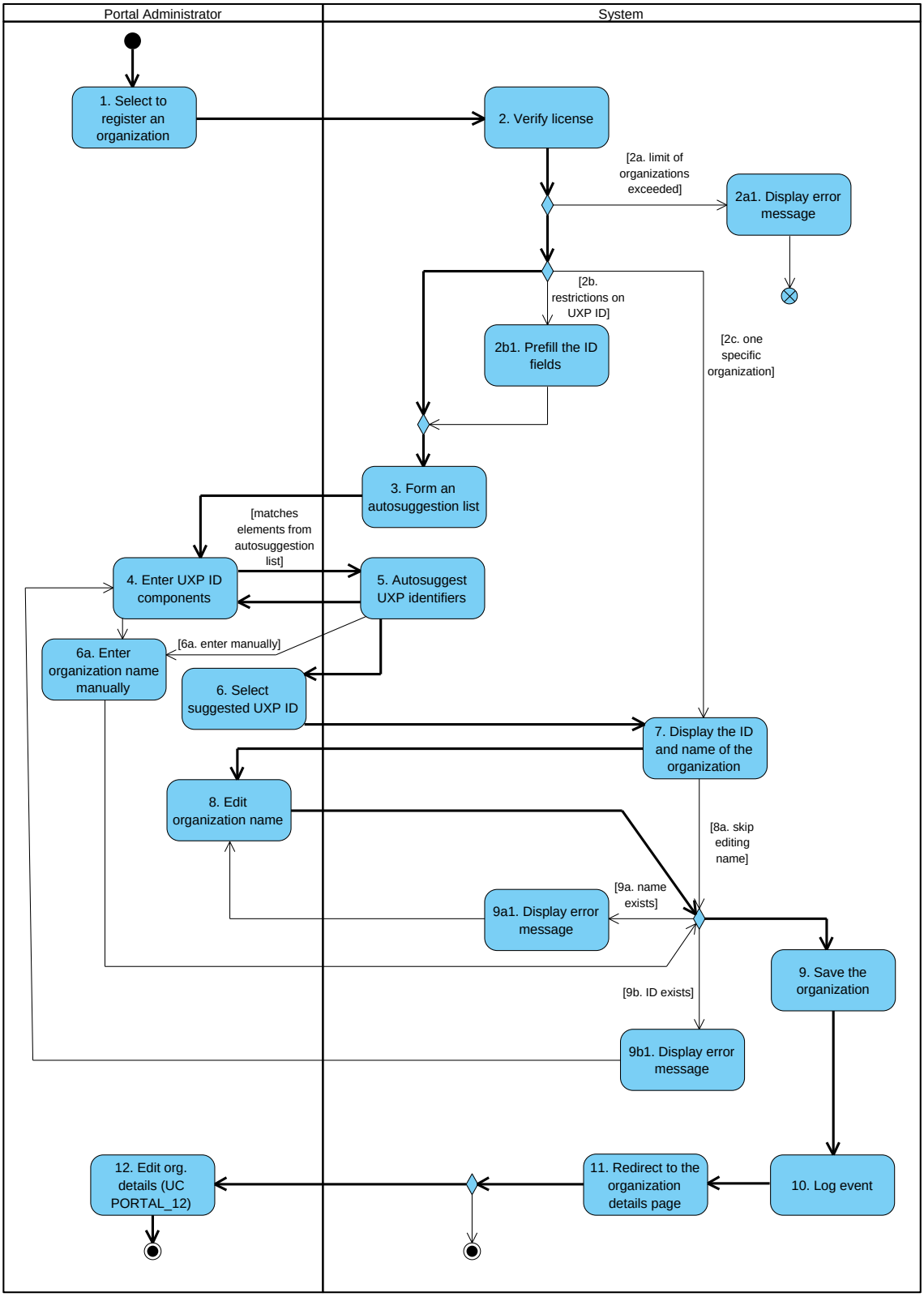

Figure 17. Activity diagram of the use case *Register Organization*

5 Discussion of Results

In this Chapter, the author describes how the output of requirements specification was validated and the lessons learned in the process. Also an insight into the future of the project is added.

5.1 Life Cycle of the Use-Case Models

Iterative development process states that requirements evolve incrementally as the problem domain becomes clearer [21]. Consequently, artifacts have to be refined throughout the project and are not entirely completed by the end of the iterations but rather stabilized.

In the Inception phase, Business Use-Case Model reached a stable state and system use cases were listed and partially detailed. In the end of the phase, these artifacts were validated to be ready for the Elaboration phase.

In Elaboration phase, use cases were further detailed and the UI prototyping began. Furthermore, use cases were fine-tuned based on the changes that were discovered during UI prototyping.

5.2 Validation of Results

Initial feedback for use case models was received from team members. This was gathered both as comments on documents and during meetings.

The biggest test for the use cases was developing the user interface prototype. According to the RUP, the UI prototypes are developed in parallel with use cases [21], therefore the use cases weren't fully detailed to avoid detailing the use cases twice. As it turned out, all the use cases required to describe the operations with different system entities were present. These operations include creating, deleting and modifying entities.

Some of the use cases had to be generalized to include them in others. For example the process of generating TLS certificates was required by multiple use cases. Generalizing use cases also gives developers a better overview of where the functionality can be implemented once and later reused.

Due to the fact that the architect of the project was actively present during the meetings, some new requirements appeared that demanded changes in the use case

descriptions. For example, the requirement that it must be possible to use TLS authentication when connecting to an LDAP database, required including TLS management use cases in the *Configure Connection to LDAP Server* use case.

During the prototyping process, new use cases were added describing how lists of data like users or services are displayed. These use cases were omitted in the inception phase to make no assumptions about the structure of the UI in the early process and keep the focus on the system functionality. The author of the Thesis decided to describe these use cases when the prototype layout was sufficiently stable. These use cases along with the corresponding view will give the developers an idea, how different functionalities are connected, as they contain references to other use cases that must be possible to access.

5.3 Lessons Learned

The external interaction designer preferred to use interviews and meetings with the project team as the primary source for UI requirements. Their method was using a storyboard where the workflows were described for each UXP Portal role. Therefore, communicating the use cases and their details described in the use case documents was the job of the author.

In addition to giving feedback to the work done by the interaction designer, the author of the Thesis drew sketches of UI to communicate the vision of layouts and logic of more complicated structures. Relatively early into the UI prototyping process it became evident that introducing the system from the very beginning can be time-consuming. Therefore, it was helpful that the project team was already familiar with the requirements so the main focus could be on user interaction. Despite the time spent on communicating the requirements , the author believes, that having a person with expertise in UX is useful to introduce new ideas to the project. This might be difficult when all project members are familiar with the previous project, in this case UXP Portal 1.0.

Working along with a UX specialist provided valuable experience both in the prototyping process and in communicating the requirements. In case no UX specialist is included in a future project and UI prototype is developed within the project team, the author would start drawing possible UI layouts sooner to try out ideas for implementing features. In UXP Portal 2.0 project, the UX and UI design are outsourced from a specialized company. UI prototyping process showed that

the requirements were sufficiently detailed by the time the external interaction designer joined the project.

The requirements specification was complicated by the lack of a direct customer. This eliminated the option to study the workflows of a certain organization. The process was made even more difficult by the fact that the product has to suit the needs of users from different cultures. Therefore, assumptions about the organizations structure had to be made to continue the work. Moreover, the team members had different opinions regarding the background and goals of the hypothetical end user. For example, discussion about how well the end user knows the working principles of the UXP infrastructure was needed to determine the level of detail of information displayed in the user interfaces. As a result, the team decided to simplify the user interfaces in order to not to overburden the users with confusing information.

In the author's opinion, developing use cases for eliciting and specifying requirements has proven to be effective so far as there have been no major changes in the use cases after the UI prototyping process. The most important lesson learned from the RUP is to keep the iterative process in mind and avoid describing too many details in the early steps of the process.

5.4 Future Work

Once the UI layout prototype is stable enough, a UI designer will join the project to specify visual design elements. After finalizing the design, developers will start implementing the functionalities. This will be the second evaluation of the use cases. It is likely that there will be changes in use cases during that process as well. According to the RUP, the requirements have to be updated by the analyst throughout the rest of the project.

It is likely that the UXP Portal will be customized for specific customers to meet the regulations of their UXP installations. For example, the regulations might state the need for a different authentication method not present in the standard version of the UXP Portal. Furthermore, the customer might require hiding or displaying some information for certain user roles.

6 Summary

In this Thesis, the author presented use-case models developed to describe the business processes around UXP Portal and to specify the functional requirements of the UXP Portal version 2.0. Use-case models were created for both business and system domains and supplemented with domain models. The system use-cases will be used for the future development process. UXP Portal is a product of Cybernetica that can be used to consume web services over UXP infrastructure. The main trial for the integrity and level of detail for the produced documents was the process of developing the first version of the UI prototype along with the external UX specialist. The use case method proved to be effective and the use cases were sufficiently detailed by the time the UX specialist engaged in the project. The next benchmark where suitability of the requirements specification will be assessed, is the process of implementing the actual system. As the RUP – the software development process of the Portal 2.0 project – is iterative, the changes in requirements are inevitable and the author's task as the project's analyst is to keep specification documents up to date when changes are applied.

References

[1] Unified eXchange Platform [Internet]; [cited 18.03.2017]. Available from: https://cyber.ee/en/e-government/uxp/. 1, 2.1, 2.1.2, 2.1.2

[2] Rational Unified Process: Best Practices for Software Development Teams [Whitepaper]; 1998 [cited 25.03.2017]. Rev 11/01. Available from: https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf. 1, 4

[3] Unified Modeling Language [Internet]; [cited 25.03.2017]. Available from: http://www.uml.org/. 1

[4] Cybernetica to deploy a secure data exchange layer similar to X-Road for Namibia [Internet]; 2014 [cited 14.04.2017]. Available from: https://cyber.ee/en/news/cybernetica-to-deploy-a-secure-data-exchange-layer-similar-to-x-road-for-namibia/. 2.1

[5] Cybernetica is deploying security technology to Haiti [Internet]; 2015 [cited 14.04.2017]. Available from: https://cyber.ee/en/news/cybernetica-is-deploying-security-technology-to-haiti/. 2.1

[6] SUNFISH project [Internet]; [cited 10.04.2017]. Available from: https://cyber.ee/en/research/research-projects/sunfish/. 2.1

[7] SecUre iNFormatIon SHaring in federated heterogeneous private clouds [Internet]; [cited 15.04.2017]. Available from: http://www.sunfishproject.eu/sunfish/the-project/. 2.1

[8] X-Road project [Internet]; [cited 15.04.2017]. Available from: https://cyber.ee/en/e-government/x-road/. 2.1

[9] Data exchange layer [Internet]; [cited 01.05.2017]. Available from: http://vm.fi/palveluvayla?p_p_id=56_INSTANCE_SSKDNE5ODInk&_56_INSTANCE_SSKDNE5ODInk_languageId=en_US. 2.1

[10] Estonia and Finland to develop the X-road together [Internet]; 2016 [cited 01.05.2017]. Available from: https://www.mkm.ee/en/news/estonia-and-finland-develop-x-road-together. 2.1

[11] Box D, Ehnebuske D, Kakivaya G, Layman A, Mendelsohn N, Nielsen HF, et al.. Simple Object Access Protocol (SOAP) 1.1 [Internet]; 2000 [cited 01.04.2017]. W3C Note. Available from: https://www.w3.org/TR/2000/NOTE-SOAP-20000508/. 2.1.1

[12] X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP [Internet]; 2013 [cited 01.04.2017]. Request for Comments 6960. Available from: https://tools.ietf.org/html/rfc6960. 2.1.2

[13] Apache CXF: An Open-Source Services Framework [Internet]; [cited 22.04.2017]. Available from: http://cxf.apache.org/. 2.4

[14] Java™ API for XML Web Services (JAX-WS) [Internet]; [cited 15.04.2017]. Available from: http://docs.oracle.com/javase/7/docs/technotes/guides/xml/jax-ws/. 2.4

[15] WSDL to Java [Internet]; [cited 22.04.2017]. Available from: http://cxf.apache.org/docs/wsdl-to-java.html. 2.4

[16] Savon - Heavy metal SOAP client [Internet]; [cited 22.04.2017]. Available from: http://savonrb.com/. 2.4

[17] SoapUI Documentation: SOAP and WSDL [Internet]; [cited 10.04.2017]. Available from: https://www.soapui.org/soap-and-wsdl/getting-started.html. 2.4

[18] ExtensionBoomerang - SOAP & REST Client [Internet]; [cited 22.04.2017]. Available from: https://chrome.google.com/webstore/detail/boomerang-soap-rest-clien/eipdnjedkpcnlmmdfdkgfpljanehloah. 2.4

[19] Generic SOAP Client [Internet]; [cited 22.04.2017]. Available from: http://www.soapclient.com/soaptest.html. 2.4

[20] Xydra - An automatic form generator for Web Services [Internet]; [cited 22.04.2017]. Available from: http://www.extreme.indiana.edu/xgws/xydra/. 2.4

[21] Kroll P, Kruchten P. The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP. Addison-Wesley Object Technology Series. Massachusetts: Pearson Education; 2003. 3, 3.1, 3.2, 5.1, 5.2

[22] Taft D. IBM Acquires Rational [Internet]; 2002 [cited 01.05.2017]. Available from: http://www.eweek.com/pc-hardware/ibm-acquires-rational. 3

[23] Kruchten P. The Rational Unified Process: An Introduction. The Addison-Wesley object technology series. Massachusetts: Addison-Wesley; 1998. 3.1, 3.3

[24] Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. Object Technology Series. Massachusetts: Addison-Wesley; 2000. 3.3, 4.1.3

[25] Roff J. UML: A Beginner's Guide. Beginner's Guide. California: McGraw-Hill Education; 2003. Available from: https://books.google.ee/books?id=ax4Ccv-hQycC. 4.2.2

[26] Cockburn A. Writing Effective Use Cases. Crystal series for software development. Michigan: Pearson Education; 2000. 4.2.3

Non-exclusive license to reproduce Thesis and make Thesis public

I, **Rebeka Mändar**,

1. herewith grant the University of Tartu a free permit (non-exclusive license) to:

   1.1 reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   1.2 make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

   UXP Portal 2.0 Functional Requirements Specification,

   supervised by **Margus Freudenthal** and **Marlon Dumas**.

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive license does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 11.05.2017