Architectural Design Document
Eindhoven, January 15, 2010

Project team:
Roy Berkeveld, 0608170
Gijs Direks, 0611093
Michael van Duijkeren, 0535368
Neal van den Eertwegh, 0610024
Dion Jansen, 0590077
Koen Kivits, 0608715
Sander Leemans, 0608896
Kevin van der Pol, 0620300
Nick van der Veeken, 0587266

Computer Science, TU/e

Project Manager:
Wilco Belgraver Thissen, 0514143
Quality Assurance Manager:
Jelle Hellings, 0592127
Senior management:
Mark van den Brand, HG 5.59
Lou Somers, HG 5.36
Advisor:
Erik Luit, HG 7.12
Customer:
Natalia Sidorova, HG 7.84

TU/e
Technische Universiteit
Eindhoven
University of Technology

Where innovation starts

Project team:
Roy Berkeveld, 0608170
Gijs Direks, 0611093
Michael van Duijkeren, 0535368
Neal van den Eertwegh, 0610024
Dion Jansen, 0590077
Koen Kivits, 0608715
Sander Leemans, 0608896
Kevin van der Pol, 0620300
Nick van der Veeken, 0587266

Computer Science, TU/e
Abstract

This document contains descriptions of the architecture of the QIS application. This application is developed for the Software Engineering Project (2IP35) at Eindhoven University of Technology. The document complies with the Architectural Design Document (ADD) from the Software Engineering Standard, as set by the European Space Agency.
# Contents

1 Introduction .................................................. 5
   1.1 Purpose .................................................. 5
   1.2 List of definitions ...................................... 5
   1.3 List of references ...................................... 5
   1.4 Overview ................................................. 6

2 System overview .............................................. 7

3 System context ................................................ 8
   3.1 External interface definition for NT-authentication ....... 8

4 System design .................................................. 10
   4.1 Design method ........................................... 10
   4.2 Decomposition description ................................ 10
   4.3 Introduction to the Django framework ..................... 12
   4.4 Integration of QIS into the Django framework ............ 16
   4.5 Internals of the QIS webserver .......................... 19
   4.6 Code structure .......................................... 20

5 Component descriptions ................................. 23
   5.1 Description method ..................................... 23
   5.2 Rights ...................................................... 25
   5.3 Authentication ........................................... 27
   5.4 Administrative objects .................................. 29
   5.5 Closing ..................................................... 31
   5.6 Copying ..................................................... 32
   5.7 Courses and course instances ............................ 33
   5.8 Study programs and target groups ....................... 34
   5.9 Employees and employment ............................... 35
   5.10 Tasks ....................................................... 37
   5.11 Assignments ............................................. 38
   5.12 Provisional versions .................................... 39
   5.13 Reports ................................................... 41

6 Feasibility and resource estimates ......................... 43

7 Requirements traceability matrix .......................... 44
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1</td>
<td>URD to ADD</td>
<td>44</td>
</tr>
<tr>
<td>7.2</td>
<td>ADD to URD</td>
<td>45</td>
</tr>
<tr>
<td>7.3</td>
<td>SRD to ADD</td>
<td>46</td>
</tr>
<tr>
<td>7.4</td>
<td>ADD to SRD</td>
<td>48</td>
</tr>
</tbody>
</table>
Document Status Sheet

Document status overview

General

Document title: Architectural Design Document
Identification: ADD-1.0.3105
Authors: Nick van der Veeken, Roy Berkeveld, Kevin van der Pol, Sander Leemans
Document status: Final

Document history

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author</th>
<th>Reason of change</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>30-11-2009</td>
<td>Kevin v.d. Pol</td>
<td>Draft</td>
</tr>
<tr>
<td>0.1</td>
<td>14-01-2010</td>
<td>Nick v.d. Veeken</td>
<td>First revision</td>
</tr>
<tr>
<td>1.0</td>
<td>15-01-2010</td>
<td>Nick v.d. Veeken</td>
<td>First version</td>
</tr>
</tbody>
</table>
Document Change Records since previous issue

General

Date: 30-11-2009
Document title: Architectural Design Document
Identification: ADD-1.0.3105

Changes

<table>
<thead>
<tr>
<th>Page</th>
<th>Paragraph</th>
<th>Reason to change</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Purpose

A formal and abstract description of QIS is given in the Software Requirements Document (SRD)[1]. This document bridges the gap between the SRD and the concrete details of the implementation of QIS, through a description of its software architecture, focusing on the various components that constitute the implementation.

1.2 List of definitions

- ADD: Architectural Design Document
- Django: A framework for web application development
- ESA: European Space Agency
- SRD: Software Requirements Document
- URD: User Requirements Document
- RDBMS: Relational Database Management System
- ORM: Object-Relational Manager

1.3 List of references


1.4 Overview

This document describes the various components that constitute the implementation of QIS. Logical models, like the class diagram for QIS and a formal description of the classes, methods and attributes are already given in the SRD [1]. This document complements the logical models with concrete architectural models.

Chapter 2 describes the design of QIS on the highest level. Chapter 3 gives a description of the protocols used to communicate with external systems. Chapter 4 describes the architectural design of QIS: It includes an introduction to the Django framework, a decomposition of QIS into components and a section on integration of these components into the framework. Chapter 5 descriptions each of the components in detail. Chapter 6 gives resource estimates for QIS, describing the resources needed to develop and operate QIS. Finally, chapter 7 contains a requirements traceability matrices, showing how the user requirements of the URD and the software requirement of the SRD are linked to the components of the ADD.
Chapter 2

System overview

For a description of QIS and the relevant background, see the User Requirements Document (URD) [2]. For a description of the environment in which QIS will operate, see the Software Requirements Document (SRD) [1].

QIS is a web-based information system. It can interact with various other information systems in its environment. Figure 2.1 shows how QIS is embedded into that environment.

Figure 2.1: QIS and its environment
Chapter 3

System context

This chapter describes the connections that QIS has with external systems. QIS must be able to operate properly without any of these connections. Any connection can be disabled independently from other connections. Currently QIS supports one connection, which is described below in section 3.1.

3.1 External interface definition for NT-authentication

The NT-authentication interface is used to authenticate users by means of (NT-username, password) pairs, which are called credentials hereafter.

Credentials are validated by attempting to log on to an LDAP server. The connection made by users to QIS requires HTTPS for security reasons: Credentials are sent over HTTPS to the webserver running QIS. The connection from QIS to the LDAP server is encrypted using the Kerberos protocol.

3.1.1 Implementation of the interface

Django offers a user authentication system that supports authentication through custom-made backends. The interface is implemented as an alternate authentication backend, which reimplements the check_password() function by performing an LDAP query to the Active Directory component of a domain controller on the domain QIS is configured for. Active Directory is essentially an LDAP server, which supports LDAP queries. Authentication is checked by attempting to log on to the LDAP server with the user-provided credentials.

To retrieve the hostname or IP-address of a domain controller, two options are provided:

1. The domain controller address is hard-coded in the configuration of QIS
2. The domain controller is located by querying the network for available domain controllers belonging to the configured domain, and selecting one of them
The external library python-ad (which has some unknown dependencies) is used if the second option is used. To allow QIS to function without these dependencies the first option is provided.

The python-ad library will query available domain controllers at most once every 5 minutes, which means that when the current domain controller becomes unavailable, QIS will refuse authentication attempts for up to 5 minutes. If the domain controller address is hard-coded, QIS will refuse authentication attempts only until the moment the controller has been recovered.
Chapter 4

System design

This chapter describes the technical aspects of the design of QIS.

4.1 Design method

QIS is implemented as a web application based on the Django framework. The framework supplies many useful components for the development of web applications. The structure and terminology of the framework determine the structure and terminology used in this description of QIS.

Section 4.2 defines the components of QIS and their dependencies. Section 4.3 offers a short introduction to the Django framework. Please refer to the Django website\(^1\) for more information on Django. Section 4.4 describes how QIS is integrated into the Django framework.

4.2 Decomposition description

The decomposition of QIS into components is based on the requirements of the URD\(^2\) and the SRD\(^1\).

Section 4.2.1 lists the components, which are further described in chapter 5. Figure 4.1 illustrates the dependencies between the components.

4.2.1 List of components

The following components are identified:

- Rights
- Authentication
- Administrative objects

\(^1\)http://www.djangoproject.com
4.2.2 Dependencies between components

Figure 4.1 illustrates the dependencies between the components of QIS. Arrows indicate a “depends on” relation between components. All components except for Authentication depend on Rights. To improve readability of the diagram, all dependencies on Rights are drawn using lighter, dotted arrows.

Figure 4.1: Components of QIS and their dependencies
4.3 Introduction to the Django framework

A Django-based application consists of models, templates and views with additional data-processing components. Because of the Django terminology of models, views and templates, Django is sometimes called an MTV framework, which is similar to the Model-View-Controller pattern.

Section 4.3.2 introduces the terms model, template and view. Section 4.3.3 describes how persistent storage is achieved. The use of programming languages in detailed in section 4.3.5. Section 4.3.4 describes how an HTTP-request is processed.

4.3.1 Version of Django

Django is an open-source project that is continuously under development. Proper functioning of QIS is only guaranteed when installed on a server running an appropriate version of Django. QIS is developed using version 1.1.1 of the Django framework.

4.3.2 Models, templates and views

4.3.2.1 Models

Models describe the logical grouping of data and functionality, like classes in the object-oriented paradigm. Django uses the word model where object-oriented paradigm uses the word class. Objects are called model instances in Django.

Because Django models are compatible with the object-oriented paradigm, the class diagram from the SRD can be mapped one-to-one into Django models. For QIS the models include all attributes as described in the class diagram of the SRD. In addition, implementation-specific attributes are added, as described in the DDD.

4.3.2.2 Templates

Templates define the looks of the user interface of a Django application. Templates consist of HTML code with special tags, which are replaced with information from specific model instances.

For example, a tag in the overview page for an employee displaying the name of the employee is inserted into the template as follows:

```html
<div class="employee_name">{{ employee.name }}</div>
```

When the template engine, an essential component of the Django MTV system, renders a template, it processes the template file and replaces all tags with the appropriate information. If in the example above the value for employee.name is “Page, L.”, the output for the rendered template will be:

```html
<div class="employee_name">Page, L.</div>
```
4.3.2.3 Views

Views define the structure of a Django application. They group units of user functionality together and they describe the possible user interactions with the models. Examples of views in QIS are the user interface components for managing employees and employment, assigning tasks, and viewing reports.

4.3.2.4 Mapping of URIs to views

Each HTTP request contains at least a URI. Django can map URIs to specific views (i.e. user functionality). It uses regular expression pattern matching to map a URI to the appropriate view. A URI may contain IDs that are used by the view, for example the following URI:

<website_root_uri>/employee/134/delete

This URI denotes that the user wants to remove the model instance for the employee with ID 134 and can be matched by the following regular expression:

r'^(employee/(?P<employee_id>.+)/delete/$'

For models the URI structure typically follows this pattern:

$model_name$/<model_instance_id>/<action>

Common actions are add, remove, edit and view, but other actions can be defined and even completely different patterns can be matched by Django, as long as it can be expressed by a regular expression.

4.3.3 Persistent storage

To enable persistent storage of data, Django ships with an Object-Relational Mapper (ORM). The ORM transparently keeps a persistent storage of all model instances created and modified in the Django application. An off-the-shelf Relational Database Management System (RDBMS) is used for the actual storage.

For the programming perspective, the ORM translates between the object-oriented way of accessing data, for example employee.employment[1].position.name, and the queries required to retrieve the correct data from the RDBMS.

Figure 4.2 illustrates the position of the ORM in the architecture of an object-oriented application that uses an RDBMS for persistent storage.

4.3.3.1 RDBMS

QIS uses MySQL 5.1 as RDBMS, but many other popular RDBMS systems are supported by Django and may be used instead. However, SQLite and MySQL 5.0 fail to execute some complex queries generated by the ORM.
Communicating with an RDBMS is a large bottleneck. To speed up the processing of requests by the web server, the ORM has a caching mechanism that reduces the number of database connections. The ORM queries related data as well: when the last name of an employee is retrieved, it is likely that the first name of the employee has to be retrieved too, so it is retrieved as well. When generating a report listing all employees of a subdepartment, the ORM retrieves relevant data of all employees in one query and keep this in cache for later use.

4.3.3.2 Relational data model

The use of an ORM enables the developers of a Django application to abstract from the relational data model used by the RDBMS both for storage and for queries. Even the creation and alteration of tables is handled by the ORM. Therefore, although an RDBMS is part of QIS, no Entity-Relation Diagram (ERD) has to be designed. The relational data model that is used by the RDBMS is generated by the ORM from the models defined in qis.app.models (see figure 4.5 in section 4.6).

4.3.4 Request processing

The way a user interacts with a web application can be described as a sequence of requests from the user to the application, each followed by response of the application. In this typical scenario the user is said to reside on the client side and the application resides on the server side.
4.3.4.1 HTTP requests and responses

A web application transfers requests and responses between client and server using the HTTP protocol. An HTTP request typically consists of a Uniform Resource Identifier (URI) and optional GET and POST data. An HTTP response consists of a status code and a message body.

4.3.4.2 Django HttpRequest and HttpResponse objects, middleware

By means of HttpRequest and HttpResponse objects, Django offers functionality for handling HTTP requests at the server side and sending back HTTP responses to the client side.

The handling process can be extended using middleware. A middleware component defines a processing step on an HttpRequest or HttpResponse object, like for example the ‘Session’ Middleware which enables session support and the ‘GZIP’ middleware which enables GZIP-compression of the HTTP response’s message body. Custom middleware can be programmed and plugged in, pre-existing middleware can be enabled or disabled and the order of application can be (re)defined. Think of it like an onion: each middleware class is a layer that wraps the view: see figure 4.3 for an illustration of the analogy. The kernel of the union is formed by the views which define the user functionality.

![Diagram of middleware layers](image)

Figure 4.3: Requests are processed through a number of successive layers of middleware

4.3.5 Programming languages

Django is built using Python, which is an interpreted high level programming language that supports object-oriented programming. Since QIS is built closely on top of Django, all code describing the models and views is written in Python. The GUI consists of a hierarchy of templates. Templates consist of HTML code with tags. To enable animation of GUI elements, Javascript is used.
4.4 Integration of QIS into the Django framework

This section describes the integration of QIS with the Django framework as described in section 4.3. This section discusses functionality that follows from the user requirements of the URD[2] and the software requirements of the SRD[1], but that is not captured in the components as discussed in chapter 5.

4.4.1 List of implemented views

This section describes the decomposition of the GUI of QIS into views, based on figure 2.2 from the URD and the rights in table 2.1 in the URD. The following views are defined:

4.4.1.1 For all users:

- Login
- Logout
- Home (view own workload)

4.4.1.2 For administrators:

- Administrative objects (for each type of administrative object)
  list, add, edit, delete (confirmation)
- External connections
  enable/disable
- Closing
- Copying

4.4.1.3 For directors of education:

- Courses
  list (includes course instances and education tasks), add, edit, delete (confirmation)
- Course instances
  add, edit, delete (confirmation)
- Education tasks
  add, edit, delete (confirmation)
- Study Programs
  list (includes target groups), add, edit, delete
• Target Groups
  add, edit, delete

4.4.1.4 For workload managers:
• Employees
  list (includes employment), add, view, edit, delete (confirmation)
• Employment
  add, edit, delete
• Tasks
  list (includes assignments), add, edit, delete (confirmation)
• Assignments
  add, edit, delete (confirmation)
• Provisional versions
  make course instances definitive (confirmation), make assignments definitive (confirmation)

4.4.1.5 For policy advisors
• Reports
  list, view

4.4.2 Administrative objects
QIS must allow for changes made in the organizational process of the client to be reflected within its interface. To allow these changes, lists like the types of education (e.g. lecture, instruction) can be modified through a straightforward user interface. The lists are collectively called administrative objects, because they will be typically maintained by an administrator of QIS. Since editing administrative objects is straightforward, the user interface for editing these objects can be generated using the Django admin interface component. Administrative objects are further discussed in section 5.4.

4.4.3 User rights management
QIS must limit the possible actions of its users according to configurable sets of rights. These sets can be created and modified by an administrator. Employees can be assigned one or more sets of rights. QIS enforces rights using model managers. Model managers are described in this
4.4.3.1 Model managers

Remember that in Django terminology, *model* is just another word for *class* as used in the object-oriented paradigm and *model instance* is just another word for *object*. Model instances are persistently stored in an RDBMS accessible through the ORM.

Imagine that all users would have the same rights, in other words, there is no user rights management. Model instances can be accessed freely without restrictions. Now to apply user rights, two kinds of restriction can be placed:

- Restrictions regarding attributes
- Restrictions regarding methods

Furthermore, restrictions can be placed on specific model instances only, instead of an all instances of a particular model.

A model manager can place all these kinds of restriction on a per-model basis. For each model a model manager is defined. Instead of accessing data and methods directly through the models, the access is passed through the respective model manager. When a user does not have sufficient rights to perform, for instance, a delete operation, the model manager blocks the operation. The great benefit of using model managers is that rights are enforced in one central place in the code and in one place only.

Access to specific attributes and methods can be restricted for each specific model and can even be limited to sets of model instances, based on the rights of the current user.

4.4.4 External connections

Connections that QIS supports can be enabled or disabled and if a connection is disabled, QIS must still be able to operate properly. To indicate if an external connection is enabled or disabled, a boolean value in the System class is defined. This boolean can be set or reset using the administrative objects view.

There are no restrictions on where the source code related to external connections is placed in the source code directory. The source code may also be spread over multiple files.

Currently, QIS implements one external connection that allows users to authenticate using their NT username and password. Section 3.1 describes this connection. The code for the connection is placed in qis.app.auth.backends. Section 5.3 describes the behaviour of QIS if the NT-authentication interface is disabled.

4.4.5 Notifications sent by QIS

If changes in workload occur, QIS must be able to send notifications by emails to the respective users. Also, passwords must be emailed to employees upon a first attempt to login to QIS if the
connection to the NT-system is disabled.

QIS provides two methods for sending email, one for sending emails to multiple addressees and another for sending emails to only one addressee. Both methods are located in the module `qis.app.mail`. This module also contains the hardcoded mail-templates used for notifying users of changes in the workload.

Whenever a provisional version of a course instance or assignment is made definitive, QIS sends notifications to the users who are authorized to view the data that has changed. A notification contains the note that something has changed and the date on which it changed. Note that the data that has changed is not included in the notification. Users are able to choose which types of notifications they would like to receive.

Initially, all users receive notifications about their own workload and all expertise group leaders receive no notifications about the workload in their expertise group. These preferences are stored as attributes of the Employee model.

### 4.4.6 Use of template tags

Templates ideally consist of HTML code and **simple tags** only. A simple tag is placeholder for information that depends on the model instances in use when processing an HTTP request. The built-in template engine of Django can replace the placeholders by actual data from model instances just before the processing of an HTTP request is complete. The processes of filling out the tags with data is called **rendering**.

Ideally, the data for each tag to be filled out should be available in the form of an associative array of strings already before the template engine starts rendering. The data is retrieved and stored in a so-called **context** object that is passed to the template engine when it starts to render.

Besides simple tags, Django also supports more complex types of tags called **template tags**. Template tags enable for data to be retrieved from model instances on the fly, while the template engine has already started rendering. The retrieval of data from model instances through template tags puts a burden on the ORM if a template features several nested iterations over model instances. Caching of query results and querying of related data is done poorly when using template tags off-the-shelf. The requirement of using only simple tags also greatly simplifies the code structure of views and templates.

Even though the use of simple tags is strongly recommended, some templates may use template tags if the number of generated queries by the ORM is below 10.

### 4.5 Internals of the QIS webservice

Figure 4.4 illustrates the internal structure of QIS. Components of the Django framework are shown in red, an external RDBMS is shown in green and the specific components for QIS are shown in blue. The RDBMS runs on the same server as QIS. Running the RDBMS on an external server introduces additional connection overhead and there may be a bandwidth bottleneck in the network connecting the RDBMS with the webservice. Running the RDBMS on the same server as
QIS reduces the costs of operation of the system and it will not harm performance because the expected server load is low.

4.6 Code structure

Django prescribes a typical code structure that is preserved in QIS. Django code is structured into a project that consists of one or more apps. QIS comes in the form of one Django project consisting of one app, named qis.app. The directory structure of QIS and a description of the most important files is given in figure 4.5. In Python, files are referenced by means of a dot notation, e.g. qis.app.models. This dot notation per definition maps directly to the directory structure of the application’s code directory, so the Python statement import qis.app.models will import the file qis/app/models.py.
Figure 4.4: Internal structure of QIS. Components of the Django framework are shown in red, an external RDBMS is shown in green and the specific components for QIS are shown in blue.
/qis
  settings.py  Configurable options like mailserver, RDBMS

/app
  __init__.py  Defines database initialization procedures
  models.py    Defines models, as in the class diagram
  modeladmins.py Defines views for specific models
  sites.py     Defines how HTTP requests (URIs) map to views
  views.py     Defines views not related to specific models

/auth
  backends.py Defines possible ways of authentication

/templatestags Defines data-retrieval functions used by templates

/media  Contains client-side media referred to by templates
/css    Contains GUI layout and style information
/img    Contains static images used in GUI
/js     Contains javascript files for GUI animation
/templates Contains HTML template definitions

Figure 4.5: Directory structure of QIS
Chapter 5

Component descriptions

Components of QIS are described in this chapter. References to relevant user requirements and software requirements are given for each component.

5.1 Description method

This section specifies how components are described.

5.1.1 Component identifier

Each component has a unique identifier.

5.1.2 Type

A component is of a certain type, which is one of user interface, connector, internal processing.

5.1.3 Purpose

A list of the software requirements that specify what is to be implemented in this component.

5.1.4 Functionality

A list of the user requirements that are implemented in this component, along with a short summary of the user requirements.

5.1.5 Dependencies

A list of components that the implementation of this component depends on.
5.1.6 Subordinates

A list of components that depend on the implementation of this component.

5.1.7 Interfaces

For user interface components: A list of references to the SRD in which the relevant parts of the prototype are described. For connectors, relevant protocols are given. Internal processing components have no interfaces.

5.1.8 Processing (optional)

For connectors and internal processing components, the description of the processing steps is given. For user interface components, standard create / read / update processing is not described, as it is implemented within the Django subsystems, any additional processing is described here.

5.1.9 Data (optional)

For most components all data that is read and written comes from the RDBMS. In that case the data description is left out. If data is imported or exported from/to other systems, as in the case of connectors, the data is described here.
5.2 Rights

Description of the interface used to grant and revoke rights. Also responsible for enforcing those rights.

5.2.1 Type

User interface component for granting and revoking rights. Also an internal processing component for enforcing and managing those rights.

5.2.2 Purpose

This component implements the following software requirements:
SCR33, 34, 40, 41, 65, 90, 100, 180.
These SCRs are only those that are implemented by Rights directly.

5.2.3 Functionality

This component implements the following user requirements:
UCR2, 3, 4, 5, 6, 8, 11, 12.

5.2.4 Dependencies

This component depends on the following components: Authentication.

5.2.5 Subordinates

The following components require this component: Administrative objects, Closing, Copying, Courses and course instances, Study programs and target groups, Employees and employment, Tasks, Assignments, Provisional versions, Reports.

5.2.6 Interfaces

For users with the rights to grant/revoke rights 2 items will be added to the interface as described in section 5.5 of the SRD. A Rightsets item which allows a user to define Rightsets and a Rights item which will allow that user to assign Rightsets to an employee.

Internally, QIS will offer the user permissions in two ways.

1. For any employee, it is possible to test presence of basic permissions. These are accessible through the Employee model (method has_right()), as well as via templates for the current employee only (variable qisperms).
2. For every model, it is possible to operate on datasets which contain only those objects the current user has permissions for. These datasets are separated in viewable, changeable and combined sets and are as Model Managers.

5.2.7 Processing

Django’s authentication system has a default permission model that consists of Users, Permissions and Groups. Users are container of identifying information. Permissions are per-model identifiers of specific actions, of which the Django administration system honours the default permissions ‘change’, ‘delete’ and ‘add’. Groups are sets of permissions. A user can be associated with any number of groups or permissions.

We require one additional model permission, ‘view’, for each of the models in QIS. QIS also holds the basic permissions as defined in the URD[2], with an additional ”Modify System Administration” basic permission. These basic permissions are represented as Groups in the Django permission model and as identifiers in QIS, set up only once during project initialization. Extending on this model in QIS, RightSets are collections of basic permissions, also represented as Groups in the Django permission model. The Django permissions associated with a RightSet’s Group is the union of all permissions of the basic permission’s Group. Finally, Rights associate RightSets with Employees and, if applicable, holds references to the object the RightSet’s basic permissions apply to, which is one of Department, Subdepartment or ExpertiseGroup. An Employee may have multiple Rights pointing to the same RightSet for differend objects.

This setup leaves the model permissions to be handled largely by Django’s authentication system, while reducing the number of objects that need to be updated when RightSets or Rights associated with Employees change. Modifications to Rights require an update of an Employee’s User’s Groups to match the RightSets that user is now assigned. Modifications to RightSets only require an update of the permissions to the RightSet’s Group.

Accessing basic permissions through the interface of Section [5.2.6] will request all RightSets associated with an employee, to see if any RightSet has set the basic permission we are interested in.

Operating on a dataset that an user has specific rights for will collect all objects scopes (the objects associated with those Rights) and see if the type of object falls under any one of those scopes (i.e. is directly related to those scopes).
5.3 Authentication

This section specifies how the Authentication component functions, specifically how both local and remote authentication function.

5.3.1 Type

This is an internal processing component for validating and managing user accounts. Acts as a connector to the NT interface. This also provides a user interface to authenticate and log out a user.

5.3.2 Purpose

This component implements the following software requirements:

5.3.3 Functionality

This component implements the following user requirements:
UCR105, 106, 117, 118.

Administrators must be able to enable and disable the external connection this component uses, and the component must still function when this connection is disabled. The connection should be able to connect to the University’s generic authentication system, and it should be possible to verify credentials with it.

5.3.4 Dependencies

This component does not depend on any other components.

5.3.5 Subordinates

The following components require this component: Rights.

5.3.6 Interfaces

The user interface component is as described in section 5.1 of the Software Requirements Document[1].
The internal NT Authentication interface follows the NT Connector interface as described in Section 3.1.
The interface to authenticate a user is provided through several UserBackend classes which integrate closely with Django’s authentication system, and are documented elsewhere. We specify two kinds of UserBackends; one for authenticating against locally stored passwords (LocalBackend), and another for direct authentication over NT (ActiveDirectoryBackend).

5.3.7 Processing

Whenever a user attempts to login, the user’s username and password are checked against a UserBackend. Depending on the current backend setting, either the ActiveDirectoryBackend or the LocalBackend is selected. Only one backend can be selected at any time, backends that are not selected will always fail authentication.

When a backend checks credentials, it first retrieves the Employee information from the database that matches the provided username. If no such Employee exists, the authentication fails. If the Employee exists, a User is retrieved from the database. If no such user exists, the user is created (copying relevant fields from the Employee and configuring rights as defined in Section 5.2). In the case of the LocalBackend, a password is generated and sent by e-mail to the e-mail address set in the Employee. Now that the user exists, the password is verified. The LocalBackend will check against a stored password which the user now knows about and can use to authenticate. The ActiveDirectoryBackend will send a request to the NT connector carrying the credentials, and authentication will only succeed if the response is True.

After authentication was successful, a session is created by Django to identify the user during consecutive requests. This session exists and a session identifier is passed via cookies until the user decides to logout. The user identifier is attached to the session to ensure that it is always possible to find out the user associated with the current request.

When the user decides to log out, the session is forcefully removed (the session identifier sent by the client no longer works) and the user will have to re-authenticate.

5.3.8 Data

The NT Authentication interface accepts two fields, username and password, which are both String fields. The result is a Boolean field. An exception may also occur.
5.4 Administrative objects

Description of the interface used to add/remove/edit administrative objects. These objects contain data on the fundamental entities in QIS that have to be modified infrequently, only when a new work planning is created, or when organizational changes take place in the department structure.

Administrative objects are: (categories in bold font)

- **Organization**
  - Departments
  - Subdepartments
  - Expertise groups

- **Temporal**
  - System years
  - Periods
  - Subperiods

- **Tasks**
  - Education types
  - Study phases
  - Positions
  - Target group names

- **User rights**
  - Right sets
  - Rights

- **System settings**
  - External connections

5.4.1 Type

User interface component for data entry and modification.

5.4.2 Purpose

This component implements the following software requirements:
SCR40, 55, 56, 70, 76, 132, 138, 139, 140, 144, 145, 146, 147, 148, 149, 150, 151.
5.4.3 Functionality

This component implements the following user requirements: UCR13, 30, 31, 32, 33, 34, 35, 36, 37, 59, 60, 61, 67, 68, 69, 76, 77, 78, 165.

5.4.4 Dependencies

This component depends on the following components: Rights.

5.4.5 Subordinates

The following components require this component: Closing, Copying.

5.4.6 Interfaces

Section 5.5 in the SRD describes the user interface for this component.
5.5 Closing

Description of the interface used to close a year.

5.5.1 Type

User interface component for data entry and modification.

5.5.2 Purpose

This component implements the following software requirements: SCR41, 65, 90, 100, 133, 134, 135, 143.

5.5.3 Functionality

This component implements the following user requirements: UCR6, 7.

5.5.4 Dependencies

This component depends on the following components: Rights, Administrative objects.

5.5.5 Subordinates

No other components require this component.

5.5.6 Interfaces

For users with the rights to modify workload information, a button “Close year” will be added to the modify subdepartment screen.

5.5.7 Processing

What appears to the user as closing a system year is actually closing a subdepartment, which is attached to a system year. When closing a subdepartment, that subdepartment’s assignments and course instances are made definitive and the provisional version will be discarded. After a subdepartment has been closed, nothing in that subdepartment may be modified. This will be enforced by the model manager. Upon closing the last open subdepartment of a subdepartment, the system year will be marked as closed by setting its “Closed” attribute to True. No user interface is provided to undo the closing of a subdepartment or a system year.
5.6 Copying

Description of the interface used to copy a year.

5.6.1 Type

User interface component for data entry and modification.

5.6.2 Purpose

This component implements the following software requirements:
SCR86, 87.

5.6.3 Functionality

This component implements the following user requirements:
UCR164, 166.

5.6.4 Dependencies

This component depends on the following components: Rights, Administrative objects.

5.6.5 Subordinates

No other components require this component.

5.6.6 Interfaces

For users with the rights to modify workload information, an interface to copy course data from a previous system year will be added to the create system year screen.

For users with the rights to modify workload information, an interface to copy workload data from a previous system year will be added to the create system year screen.

5.6.7 Processing

Copying data from a previous system year will copy the data of all subdepartments from that system year, so the copying is not limited to a single subdepartment.
5.7 Courses and course instances

Description of the interface used to add/remove/edit course and course instance data.

5.7.1 Type

User interface component for data entry and modification.

5.7.2 Purpose

This component implements the following software requirements:
SCR6, 7, 8, 10, 13, 14, 15, 16, 17, 18, 20, 21, 30, 42, 43, 44, 45, 46, 47, 48, 49, 50, 52, 53, 55, 56, 69, 70, 72, 75, 76, 77, 82, 83, 103, 104, 105, 106, 109, 152, 153.

5.7.3 Functionality

This component implements the following user requirements:
UCR14, 15, 16, 17, 19, 21, 22, 23, 24, 25, 26, 29, 30, 31, 32.

Directors of Education, with restriction to their subdepartment, can add/remove/edit courses and course instances. Furthermore, workload managers and directors of education can add/remove/edit education tasks. Study phases and education types are also part of this component.

Study Phases are designed as a list, and adding/removing/editing of them is done through the administrative objects list.

5.7.4 Dependencies

This component depends on the following components: Rights, Provisional versions.

5.7.5 Subordinates

The following components require this component: Study programs and target groups, Assignments.

5.7.6 Interfaces

Section 5.3 in the SRD describes the user interface for this component. One specific modification made in agreement with the client is to display one row per course instance and to display the course instances of one course under each other.
5.8 Study programs and target groups

Description of the interface used to add/remove/edit study programs and target groups, and assign courses-instances to target groups.

5.8.1 Type

User interface component for data entry and modification.

5.8.2 Purpose

This component implements the following software requirements: SCR136, 141, 142, 155, 156, 157, 158, 159, 160, 161, 162, 163.

5.8.3 Functionality

This component implements the following user requirements: UCR18, 96, 97, 98, 99, 100, 101.

Directors of Education, with restriction to their subdepartment, can add/remove/edit study programs and target groups. Furthermore, directors of education can assign courses to specific target groups of study programs.

Study Phases are designed as a list, and adding/removing/editing of them is done through the administrative objects list.

5.8.4 Dependencies

This component depends on the following components: Rights, Courses and course instances.

5.8.5 Subordinates

No other components require this component.

5.8.6 Interfaces

The user interface for this component is not specifically described in the SRD. The user interface will look similar to that of the Employees and Employment component, containing a list of study programs instead. Each study program can be expanded, after which its target groups are shown. Each target groups edit interface closely resembles the interface as shown in figure 5.15 in the SRD.
5.9 Employees and employment

Description of the interface used to add/remove/edit employee and employment data.

5.9.1 Type

User interface component for data entry and modification.

5.9.2 Purpose

This component implements the following software requirements:
SCR22, 23, 24, 27, 28, 81, 166, 168, 169.

5.9.3 Functionality

This component implements the following user requirements:
UCR6, 9, 10, 48, 49, 50, 52, 53, 54, 55, 58, 72, 129, 130, 131, 132, 141, 142.

Workload managers, with restriction to their subdepartment, can add/remove/edit employees and their employment. Each employee is a user that has an account that can be created and removed by either an administrator or a workload manager, for the latter this is again restricted to their subdepartment.

Workload managers can view the number of hours available for planning for all employees of their subdepartment (the subdepartment restriction holds for all following statements too). Workload managers can change the employment of an employee, which means the amount of ftes, position, start and end date, expertise group and the fte-ratios the employee has to spend on education tasks, research projects and management projects.

5.9.4 Dependencies

This component depends on the following components: Rights, Provisional versions.

5.9.5 Subordinates

The following components require this component: Assignments, Reports.

5.9.6 Interfaces

Section 5.2 in the SRD describes the user interface for this component. One specific modification made in agreement with the client is to display one row per employment and to display the employment of one employee under each other.
5.9.7 Processing

Each employee is a user of QIS and therefore needs to have an account. A workload manager is allowed to create accounts with right ‘view own workload’. Employees that are leader of an expertise group have to be given the right ‘view expertise group workload’ by an administrator.
5.10 Tasks

Description of the interface used to add/remove/edit types of tasks.

5.10.1 Type

User interface component for data entry and modification.

5.10.2 Purpose

This component implements the following software requirements:

5.10.3 Functionality

This component implements the following user requirements:
UCR28, 79, 80, 81, 83, 86, 87, 88, 90, 91, 92, 93, 95, 102, 103, 104.

Workload managers, with restriction to their subdepartment, can add, edit or remove tasks of the following categories: Research project, Management project, Education project, Other task, Leave and enter the number of hours this type of task typically amounts to. They can also assign employees a task of one of these type of tasks.

5.10.4 Dependencies

This component depends on the following components: Rights, Provisional versions.

5.10.5 Subordinates

The following components require this component: Assignments, Reports.

5.10.6 Interfaces

Section 5.4 in the SRD[1] describes the user interface for this component.
5.11 Assignments

Description of the interface used to assign tasks to employees.

5.11.1 Type

User interface component for data entry and modification.

5.11.2 Purpose

This component implements the following software requirements:
SCR17, 72, 82, 83, 103, 105, 106, 109, 110, 111, 112, 118, 123, 124, 125, 126, 128,
129, 130, 188.

5.11.3 Functionality

This component implements the following user requirements:
UCR27, 28, 82, 84, 89, 94, 102, 103, 104, 158, 159.

Workload managers, with restriction to their subdepartment, can view, assign or unassign a number
of hours of certain tasks or long planned leaves to employees. They can also specify for their
subdepartment whether hours assigned to a research project are funded from inside or outside its
subdepartment.

Workload managers are able to enter manual corrections of the outcome of a formula for an
assignment and an empty formula always evaluates to 0.

5.11.4 Dependencies

This component depends on the following components: Rights, Courses and course instances,
Employees and employment, Tasks, Provisional versions.

5.11.5 Subordinates

No other components require this component.

5.11.6 Interfaces

Section 5.4 in the SRD describes the user interface for this component.
5.12 Provisional versions

5.12.1 Type

User interface component for making provisional information definitive.

5.12.2 Purpose

This component implements the following software requirements: SCR1, 19, 36, 54, 84, 85, 112, 124, 127, 154, 163.

See section 2.7.4 of the SRD for details on the relevant software requirements.

5.12.3 Functionality

This component implements the following user requirements: UCR38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 120.

Course instances and assignments can be edited on a provisional basis. QIS provides a method to make a set of assignment definitive and a method to make a set of course instances definitive. Making a provisional version definitive overwrites any previous definitive versions. See section 2.2.8 of the URD for more details on the relevant user requirement.

5.12.4 Dependencies

This component depends on the following components: Rights.

5.12.5 Subordinates

The following components require this component: Courses and course instances, Employees and employment, Tasks, Assignments.

5.12.6 Interfaces

For users with the rights to make course information definitive, a button “Courses definitive” is added to the courses view. For users with the rights to make assignments definitive, a button “Assignments definitive” is added to the employees view.

5.12.7 Processing

In the models of courses instances and assignments, a boolean attribute definitive is added. For each course instance and for each assignment, two model instances are stored in the persistent storage. The model instances that are definitive have the boolean attribute definitive set to
true, the model instances that are provisional have the boolean attribute definitive set to false. When making a provisional version definitive, the old definitive version will be removed and the current provisional version will be copied and marked as the new definitive version by setting the attribute definitive to true.

The following design decisions have been made:

1. Classes CourseInstance and Assignment have an attribute definitive, which signifies that the course instance or assignment is definitive.

2. Class Task has a boolean attribute to_be_removed, which signifies that the task is to be removed when making the assignments definitive. Instead of removing tasks directly, this boolean will be set to true when removing a task. Otherwise, the references from definitive assignments have to be updated as well, violating the principle that the definitive version is stable until a new provisional version is made definitive. It will, however, be possible to change attributes of a task which will be directly visible in the definitive version of assignments. However, these attributes will not affect the workload of employees directly. Therefore this is not considered a problem.

3. Classes Subperiod, Employee and Employment have the same problem as Task, but no similar solution is used for these classes. The reason for this is that it is assumed that these will not be modified on a provisional basis. For example, when an employee is laid off, it will be intuitive to the user that any assignments associated with his employment are removed along with his employment, whether these assignments are definitive or not.

4. Class EducationTask has two many-to-one references to the course instance it belongs to, one for the provisional object and one for the related definitive object. These references are both optional.
5.13 Reports

Description of the interface and back-end used to generate reports.

5.13.1 Type

User interface component for selecting data to report on and a system for displaying reports of the selected data.

5.13.2 Purpose

This component implements the following software requirements: SCR23, 37, 58, 88, 91, 92, 93, 94, 95, 101, 191.

5.13.3 Functionality

This component implements the following user requirements: UCR20, 51, 56, 57, 62, 63, 64, 65, 70, 71, 72, 73, 74, 75, 167, 170, 171, 172, 173, 174, 175.

Workload managers, with restriction to their subdepartment, can view reports of various kinds. Reports can be printed using the print functionality of the browser. The report can also be exported to CSV-format using a button on the report page.

Employees are able to view their own workload and the other employees that have tasks in courses that the employee has one or more tasks in himself/herself.

5.13.4 Dependencies

This component depends on the following components: Rights, Employees and employment, Tasks.

5.13.5 Subordinates

No other components require this component.

5.13.6 Interfaces

Section 5.6 in the SRD describes the user interface for this component. The reports are generated using templates.
5.13.7 Processing

The processing for reports consists of selecting the right entities that need to be displayed using templates.
Chapter 6

Feasibility and resource estimates

This chapter gives an estimation of the computer resources need to develop and operate QIS.

The requirements for development of QIS are:

- **CPU**: \( \geq 1.0 \text{ Ghz x86 or equivalent} \\
  \text{Memory}: \geq 1 \text{ GB RAM} \\
  \text{Hard disk}: \geq 1 \text{ GB free on hard disk} \\
  \text{Operating system}: \text{Windows XP, Vista or 7; Linux} \\
  \text{Software}: \text{Python 2.4 or higher, Django 1.1.1, Internet Explorer 7 or 8, Firefox 3.5, Email client, RDBMS supported by Django}

The requirements for operating QIS are:

- **Server side:**
  - **CPU**: \( \geq 2.0 \text{ Ghz x86 or equivalent} \\
  - **Memory**: \( \geq 2 \text{ GB RAM} \\
  - **Hard disk**: \( \geq 4 \text{ GB free on hard disk} \\
  - **Operating system**: *nix based \\
  - **Software**: Python 2.4 or higher, Django 1.1.1, Webserver supporting WSGI applications (e.g. Apache 2), RDBMS supported by Django

- **Client side:**
  - **CPU**: \( \geq 1.0 \text{ Ghz x86 or equivalent} \\
  - **Memory**: \( \geq 512 \text{ MB RAM} \\
  - **Operating system**: Any supporting the software \\
  - **Software**: Internet Explorer 7 or 8; Firefox 3.5, Email client
Chapter 7

Requirements traceability matrix

The requirements of the URD and SRD encompass more functionality than is implemented in the current version of QIS. The requirements that are must-haves in the URD are all implemented, some should-haves are also implemented. None of the could-haves and would-haves are implemented. This causes some requirements to be listed as Not matched in the matrices.

7.1 URD to ADD

<table>
<thead>
<tr>
<th>User</th>
<th>Component</th>
<th>User</th>
<th>Component</th>
<th>User</th>
<th>Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Not matched</td>
<td>60</td>
<td>Administrative</td>
<td>119</td>
<td>Not matched</td>
</tr>
<tr>
<td>2</td>
<td>Rights</td>
<td>61</td>
<td>Administrative</td>
<td>120</td>
<td>Notifications, Provisional versions</td>
</tr>
<tr>
<td>3</td>
<td>Rights</td>
<td>62</td>
<td>Reports</td>
<td>121</td>
<td>Notifications</td>
</tr>
<tr>
<td>4</td>
<td>Rights</td>
<td>63</td>
<td>Reports</td>
<td>122</td>
<td>Notifications</td>
</tr>
<tr>
<td>5</td>
<td>Rights</td>
<td>64</td>
<td>Reports</td>
<td>123</td>
<td>Notifications</td>
</tr>
<tr>
<td>6</td>
<td>Employees, Rights, Closing</td>
<td>65</td>
<td>Reports</td>
<td>124</td>
<td>Not matched</td>
</tr>
<tr>
<td>7</td>
<td>Closing</td>
<td>66</td>
<td>Not matched</td>
<td>125</td>
<td>Notifications</td>
</tr>
<tr>
<td>8</td>
<td>Rights</td>
<td>67</td>
<td>Administrative</td>
<td>126</td>
<td>Notifications</td>
</tr>
<tr>
<td>9</td>
<td>Employees</td>
<td>68</td>
<td>Administrative</td>
<td>127</td>
<td>Not matched</td>
</tr>
<tr>
<td>10</td>
<td>Employees</td>
<td>69</td>
<td>Administrative</td>
<td>128</td>
<td>Not matched</td>
</tr>
<tr>
<td>11</td>
<td>Rights</td>
<td>70</td>
<td>Reports</td>
<td>129</td>
<td>Employees</td>
</tr>
<tr>
<td>12</td>
<td>Rights</td>
<td>71</td>
<td>Reports</td>
<td>130</td>
<td>Employees</td>
</tr>
<tr>
<td>13</td>
<td>Administrative</td>
<td>72</td>
<td>Employees, Reports</td>
<td>131</td>
<td>Employees</td>
</tr>
<tr>
<td>14</td>
<td>Courses</td>
<td>73</td>
<td>Reports</td>
<td>132</td>
<td>Employees</td>
</tr>
<tr>
<td>15</td>
<td>Courses</td>
<td>74</td>
<td>Reports</td>
<td>133</td>
<td>Courses</td>
</tr>
<tr>
<td>16</td>
<td>Courses</td>
<td>75</td>
<td>Reports</td>
<td>134</td>
<td>Courses</td>
</tr>
<tr>
<td>17</td>
<td>Courses</td>
<td>76</td>
<td>Administrative</td>
<td>135</td>
<td>Courses</td>
</tr>
<tr>
<td>18</td>
<td>Study programs</td>
<td>77</td>
<td>Administrative</td>
<td>136</td>
<td>Courses</td>
</tr>
<tr>
<td>19</td>
<td>Courses</td>
<td>78</td>
<td>Administrative</td>
<td>137</td>
<td>Not matched</td>
</tr>
<tr>
<td>20</td>
<td>Reports</td>
<td>79</td>
<td>Tasks</td>
<td>138</td>
<td>Not matched</td>
</tr>
<tr>
<td>21</td>
<td>Courses</td>
<td>80</td>
<td>Tasks</td>
<td>139</td>
<td>Not matched</td>
</tr>
<tr>
<td>22</td>
<td>Courses</td>
<td>81</td>
<td>Tasks</td>
<td>140</td>
<td>Not matched</td>
</tr>
<tr>
<td>23</td>
<td>Courses</td>
<td>82</td>
<td>Assignments</td>
<td>141</td>
<td>Employees</td>
</tr>
<tr>
<td>24</td>
<td>Courses</td>
<td>83</td>
<td>Tasks</td>
<td>142</td>
<td>Employees</td>
</tr>
</tbody>
</table>
### 7.2 ADD to URD

<table>
<thead>
<tr>
<th>Component</th>
<th>User</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrative</td>
<td>13, 30, 31, 32, 33, 34, 35, 36, 37, 59, 60, 61, 67, 68, 69, 76, 77, 78, 165</td>
</tr>
<tr>
<td>Employees</td>
<td>6, 9, 10, 48, 49, 50, 52, 53, 54, 55, 58, 72, 129, 130, 131, 132, 141, 142</td>
</tr>
<tr>
<td>Authentication</td>
<td>105, 106, 117, 118</td>
</tr>
<tr>
<td>Tasks</td>
<td>14, 15, 16, 17, 19, 21, 22, 23, 24, 25, 26, 29, 30, 31, 32, 133, 134, 135, 136, 143, 144</td>
</tr>
<tr>
<td>Courses</td>
<td>28, 79, 80, 81, 83, 86, 87, 88, 90, 91, 92, 93, 95, 102, 103, 104</td>
</tr>
<tr>
<td>Study programs</td>
<td>18, 96, 97, 98, 99, 100, 101</td>
</tr>
<tr>
<td>Assignments</td>
<td>22, 28, 82, 84, 89, 90, 102, 103, 104, 158, 159</td>
</tr>
<tr>
<td>Notifications</td>
<td>120, 121, 122, 123, 125, 126</td>
</tr>
<tr>
<td>Provisional versions</td>
<td>38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 120</td>
</tr>
</tbody>
</table>
### 7.3 SRD to ADD

<table>
<thead>
<tr>
<th>Software</th>
<th>Component</th>
<th>Software</th>
<th>Component</th>
<th>Software</th>
<th>Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Provisional versions</td>
<td>65</td>
<td>Rights,</td>
<td>129</td>
<td>Assignments</td>
</tr>
<tr>
<td></td>
<td>Not matched</td>
<td>66</td>
<td>Not matched</td>
<td>130</td>
<td>Assignments</td>
</tr>
<tr>
<td>3</td>
<td>Not matched</td>
<td>67</td>
<td>Not matched</td>
<td>131</td>
<td>Not matched</td>
</tr>
<tr>
<td>4</td>
<td>Not matched</td>
<td>68</td>
<td>Not matched</td>
<td>132</td>
<td>Administrative</td>
</tr>
<tr>
<td>5</td>
<td>Not matched</td>
<td>69</td>
<td>Courses</td>
<td>133</td>
<td>Closing</td>
</tr>
<tr>
<td>6</td>
<td>Courses</td>
<td>70</td>
<td>Administrative, Courses</td>
<td>134</td>
<td>Closing</td>
</tr>
<tr>
<td></td>
<td>Not matched</td>
<td>71</td>
<td>Courses</td>
<td>135</td>
<td>Closing</td>
</tr>
<tr>
<td>8</td>
<td>Courses</td>
<td>72</td>
<td>Tasks,</td>
<td>136</td>
<td>Study programs</td>
</tr>
<tr>
<td></td>
<td>Courses</td>
<td>77</td>
<td>Courses</td>
<td>137</td>
<td>Not matched</td>
</tr>
<tr>
<td>10</td>
<td>Courses</td>
<td>74</td>
<td>Not matched</td>
<td>138</td>
<td>Administrative</td>
</tr>
<tr>
<td>11</td>
<td>Not matched</td>
<td>75</td>
<td>Courses</td>
<td>139</td>
<td>Administrative</td>
</tr>
<tr>
<td>12</td>
<td>Not matched</td>
<td>76</td>
<td>Administrative, Courses</td>
<td>140</td>
<td>Administrative</td>
</tr>
<tr>
<td>13</td>
<td>Courses</td>
<td>79</td>
<td>Courses</td>
<td>141</td>
<td>Study programs</td>
</tr>
<tr>
<td>14</td>
<td>Courses</td>
<td>80</td>
<td>Not matched</td>
<td>142</td>
<td>Study programs</td>
</tr>
<tr>
<td>15</td>
<td>Courses</td>
<td>81</td>
<td>Not matched</td>
<td>143</td>
<td>Closing</td>
</tr>
<tr>
<td>16</td>
<td>Courses, Assignments</td>
<td>82</td>
<td>Tasks,</td>
<td>144</td>
<td>Administrative</td>
</tr>
<tr>
<td>17</td>
<td>Employees</td>
<td>83</td>
<td>Courses, Assignments</td>
<td>145</td>
<td>Administrative</td>
</tr>
<tr>
<td>19</td>
<td>Provisional versions</td>
<td>84</td>
<td>Notifications, Provisional versions</td>
<td>146</td>
<td>Administrative</td>
</tr>
<tr>
<td>20</td>
<td>Courses</td>
<td>85</td>
<td>Notifications, Provisional versions</td>
<td>147</td>
<td>Administrative</td>
</tr>
<tr>
<td>21</td>
<td>Courses</td>
<td>86</td>
<td>Copying</td>
<td>148</td>
<td>Administrative</td>
</tr>
<tr>
<td>22</td>
<td>Employees</td>
<td>87</td>
<td>Copying</td>
<td>149</td>
<td>Administrative</td>
</tr>
<tr>
<td>23</td>
<td>Employees, Reports</td>
<td>88</td>
<td>Reports</td>
<td>150</td>
<td>Administrative</td>
</tr>
<tr>
<td>24</td>
<td>Employees</td>
<td>89</td>
<td>Not matched</td>
<td>151</td>
<td>Administrative</td>
</tr>
<tr>
<td>25</td>
<td>Notifications</td>
<td>90</td>
<td>Rights,</td>
<td>152</td>
<td>Courses</td>
</tr>
<tr>
<td>26</td>
<td>Not matched</td>
<td>91</td>
<td>Provisional versions</td>
<td>153</td>
<td>Courses</td>
</tr>
<tr>
<td>27</td>
<td>Employees</td>
<td>92</td>
<td>Reports</td>
<td>154</td>
<td>Provisional versions</td>
</tr>
<tr>
<td>28</td>
<td>Employees</td>
<td>93</td>
<td>Reports</td>
<td>155</td>
<td>Study programs</td>
</tr>
<tr>
<td>29</td>
<td>Not matched</td>
<td>94</td>
<td>Reports</td>
<td>156</td>
<td>Study programs</td>
</tr>
<tr>
<td>30</td>
<td>Courses</td>
<td>95</td>
<td>Reports</td>
<td>157</td>
<td>Study programs</td>
</tr>
<tr>
<td>31</td>
<td>Not matched</td>
<td>96</td>
<td>Not matched</td>
<td>158</td>
<td>Study programs</td>
</tr>
<tr>
<td>32</td>
<td>Rights</td>
<td>97</td>
<td>Not matched</td>
<td>159</td>
<td>Study programs</td>
</tr>
<tr>
<td>33</td>
<td>Rights</td>
<td>98</td>
<td>Rights</td>
<td>160</td>
<td>Study programs</td>
</tr>
</tbody>
</table>

46
<table>
<thead>
<tr>
<th>Page</th>
<th>Description</th>
<th>Value</th>
<th>Value</th>
<th>Value</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>Rights</td>
<td>Rights</td>
<td>Not matched</td>
<td>Study programs</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>Authentication</td>
<td>Authentication</td>
<td>Not matched</td>
<td>Study programs, Provisional versions</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>Provisional versions</td>
<td>Provisional versions</td>
<td>Rights, Closing</td>
<td>Not matched</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>Reports</td>
<td>Reports</td>
<td>Not matched</td>
<td>Employees</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>Not matched</td>
<td>Not matched</td>
<td>Tasks, Courses, Assignments</td>
<td>Authentication</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>Administrative, Rights</td>
<td>Administrative, Rights</td>
<td>Tasks, Courses, Assignments</td>
<td>Employees, Authentication</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>Rights, Closing</td>
<td>Rights, Closing</td>
<td>Tasks, Courses, Assignments</td>
<td>Employees, Authentication</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>Courses</td>
<td>Courses</td>
<td>Tasks, Courses, Assignments</td>
<td>Notifications</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>Courses</td>
<td>Courses</td>
<td>Not matched</td>
<td>Authentication</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>Courses</td>
<td>Courses</td>
<td>Not matched</td>
<td>Authentication</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>Courses</td>
<td>Courses</td>
<td>Tasks, Courses, Assignments</td>
<td>Not matched</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>Courses</td>
<td>Courses</td>
<td>Tasks, Assignments</td>
<td>Not matched</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>Courses</td>
<td>Courses</td>
<td>Tasks, Assignments</td>
<td>Not matched</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>Courses</td>
<td>Courses</td>
<td>Tasks, Assignments, Provisional versions</td>
<td>Not matched</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>Courses</td>
<td>Courses</td>
<td>Not matched</td>
<td>Not matched</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>Courses</td>
<td>Courses</td>
<td>Tasks</td>
<td>Not matched</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>Not matched</td>
<td>Not matched</td>
<td>Tasks</td>
<td>Not matched</td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>Courses</td>
<td>Courses</td>
<td>Tasks</td>
<td>Rights</td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>Courses</td>
<td>Courses</td>
<td>Tasks</td>
<td>Notifications</td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>Provisional versions</td>
<td>Provisional versions</td>
<td>Tasks, Assignments</td>
<td>Notifications</td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>Administrative, Courses</td>
<td>Administrative, Courses</td>
<td>Tasks</td>
<td>Not matched</td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>Administrative, Courses</td>
<td>Administrative, Courses</td>
<td>Tasks</td>
<td>Not matched</td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>Not matched</td>
<td>Not matched</td>
<td>Tasks</td>
<td>Not matched</td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>Reports</td>
<td>Reports</td>
<td>Tasks</td>
<td>Not matched</td>
<td></td>
</tr>
<tr>
<td>59</td>
<td>Not matched</td>
<td>Not matched</td>
<td>Tasks, Assignments</td>
<td>Not matched</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>Not matched</td>
<td>Not matched</td>
<td>Tasks, Assignments, Provisional versions</td>
<td>Assignments</td>
<td></td>
</tr>
<tr>
<td>61</td>
<td>Not matched</td>
<td>Not matched</td>
<td>Tasks, Assignments</td>
<td>Not matched</td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>Not matched</td>
<td>Not matched</td>
<td>Tasks, Assignments</td>
<td>Not matched</td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>Not matched</td>
<td>Not matched</td>
<td>Provisional versions Assignments</td>
<td>Reports</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>Not matched</td>
<td>Not matched</td>
<td>Assignments</td>
<td>Not matched</td>
<td></td>
</tr>
</tbody>
</table>
## 7.4 ADD to SRD

<table>
<thead>
<tr>
<th>Component</th>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrative</td>
<td>40, 55, 56, 70, 76, 132, 138, 139, 140, 144, 145, 146, 147, 148, 149, 150, 151</td>
</tr>
<tr>
<td>Employees</td>
<td>22, 23, 24, 27, 28, 81, 166, 168, 169</td>
</tr>
<tr>
<td>Authentication</td>
<td>35, 167, 168, 169, 171, 172</td>
</tr>
<tr>
<td>Tasks</td>
<td>72, 82, 83, 103, 104, 105, 106, 109, 110, 111, 112, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 125, 126</td>
</tr>
<tr>
<td>Courses</td>
<td>6, 7, 8, 10, 13, 14, 15, 16, 17, 18, 20, 21, 30, 42, 43, 44, 45, 46, 47, 48, 49, 50, 52, 53, 55, 56, 69, 70, 72, 75, 76, 77, 82, 83, 103, 104, 105, 106, 109, 152, 153</td>
</tr>
<tr>
<td>Study programs</td>
<td>136, 141, 142, 155, 156, 157, 158, 159, 160, 161, 162, 163</td>
</tr>
<tr>
<td>Assignments</td>
<td>17, 72, 82, 83, 103, 105, 106, 109, 110, 111, 112, 118, 123, 124, 125, 126, 128, 129, 130, 188</td>
</tr>
<tr>
<td>Notifications</td>
<td>25, 84, 85, 170, 181, 182, 183</td>
</tr>
<tr>
<td>Provisional versions</td>
<td>1, 19, 36, 54, 84, 85, 112, 124, 127, 154, 163</td>
</tr>
<tr>
<td>Rights</td>
<td>33, 34, 40, 41, 65, 90, 100, 180</td>
</tr>
<tr>
<td>Closing</td>
<td>41, 65, 90, 100, 133, 134, 135, 143</td>
</tr>
<tr>
<td>Copying</td>
<td>86, 87</td>
</tr>
<tr>
<td>Reports</td>
<td>23, 37, 58, 88, 91, 92, 93, 94, 95, 101, 191</td>
</tr>
</tbody>
</table>