D3.1 Architecture Design Document

Due date of deliverable: 01-06-2007  
Actual submission date: 24-08-2007

Start date of project: 01-12-2005  
Duration: 4 years

LOGICACMG

Revision 1.2
Project Deliverable Report

Deliverable number: D3.1 Architecture Design Document

Work Package: Work Package 3
Task

Date of delivery
Contractual: 01-06-2007
Actual: 24-08-2007

Code name: Version: 1.2
Draft ☐ Final ☑

Type of deliverable: Report

Security (distribution level): Public

Contributors
Helena Battle (FBM-UPF), Phil Beauvoir (University of Bolton), Michele Dicerto (ILABS), Arne Koesling (UHANN), Ruud Lemmers (LOGICACMG), Harrie Martens (OUNL), Ayman Moghniegh (FMB-UPF), Naiara Sacristán (Altran SDB), Miguel Arjona Villanueva (Altran SDB), Hubert Vogten (OUNL), Rob Koper (OUNL), Francisco Javier Arenales (Altran SDB), Eleni Boursinou (INSEAD), Christian Glahn (OUNL), Eelco Herder (UHANN), Pythagoras Karampiperis (CERTH/ITI), Marco Luccini (ILABS), Jocelyn Manderveld (SURF), Carlos Méndez (Altran SDB), Judith Schoonenboom (UvA), Henk Sligte (UvA), Marcus Specht (OUNL), Panagiotis Zervas (CERTH/ITI)

Authors (Partner)
Ruud Lemmers (LOGICACMG), Phil Beauvoir (University of Bolton), Rob Koper (OUNL) and Christian Glahn (OUNL)

Contact Person
Ruud Lemmers (LOGICACMG)

WP/Task responsible
Ruud Lemmers (LOGICACMG)

EC Project Officer
Hans-Juergen Westhoff

Abstract
This deliverable shows the following:
1. Communicates the architectural decisions by showing models of the system from a number of different viewpoints. These viewpoints are the Use case view, Logical view, Implementation view and Deployment view.
2. Provides guidelines for developers who wish to contribute to the integrated system.
3. The TENCompetence domain model.
4. The high and mid level requirements for the TENCompetence Infrastructure. Both general functional requirements from a user’s perspective and non-functional requirements for the infrastructure are described.

Keywords List
Architecture, Design models, Data model, Developer guidelines, domain model, requirements, actors, use cases
# Table of contents

1. Executive summary .......................................................................................................................... 7

2. Introduction ........................................................................................................................................... 9

   2.1. Aims and Objectives .................................................................................................................. 9
   2.2. Road to this architecture .......................................................................................................... 9
   2.3. Reading guide ............................................................................................................................ 10
   2.4. Acronyms .................................................................................................................................... 10

3. Use case view ......................................................................................................................................... 11

   3.1. Introduction ............................................................................................................................... 11
   3.2. Use case: Study for new function / job ..................................................................................... 12

4. Logical view ......................................................................................................................................... 15

   4.1. Analysis class diagram: Study for new job .............................................................................. 15
   4.1.1. Details of the boundary, controller and entity classes ......................................................... 16
   4.2. Design Packages ....................................................................................................................... 17
   4.2.1. TENCompetence Client Wire Frames ............................................................................... 22
   4.3.1. Introduction .......................................................................................................................... 22
   4.3.2. Editors and Views .................................................................................................................. 23
   4.3.3. My Action Plans ................................................................................................................... 24
   4.3.4. Action Plan Wizard ............................................................................................................... 24
   4.3.5. Defining the Goal of the Action Plan .................................................................................... 24
   4.3.6. New Action Plan .................................................................................................................... 26
   4.3.7. Action Plan ............................................................................................................................ 26
   4.3.8. Activity Editor ......................................................................................................................... 30
   4.3.9. Competence Development Program Editor ........................................................................ 31
   4.3.10. Agent ..................................................................................................................................... 32
   4.3.11. Persons ................................................................................................................................... 33
   4.3.12. Rating .................................................................................................................................... 33
   4.3.13. Support ................................................................................................................................. 34

4.4. TENCompetence Client Data Model ........................................................................................... 34

   4.4.1. Introduction .......................................................................................................................... 34
   4.4.2. Extension point for Data model ........................................................................................... 35
   4.4.3. Endpoints .................................................................................................................................. 36
   4.4.4. Data Model ............................................................................................................................... 37
   4.4.5. Implementation of the client Data Model ............................................................................. 39
   4.4.6. Data Dictionary ....................................................................................................................... 40

5. Implementation view ............................................................................................................................ 47

   5.1. DB data model .............................................................................................................................. 47
   5.1.1. Introduction .......................................................................................................................... 47
   5.1.2. Database Description ............................................................................................................ 47
   5.1.3. Descriptions of tables ........................................................................................................... 50

   5.2. Conduits Framework .................................................................................................................. 51
   5.2.1. Description ........................................................................................................................... 51
   5.2.2. Sharing Conduits .................................................................................................................. 51
   5.2.3. Co-ordination/Subscription conduits ................................................................................. 55

   5.3. Event Model ............................................................................................................................... 57

   5.4. Plug-in architecture of Eclipse .................................................................................................. 59
   5.4.1. Background ............................................................................................................................ 59
   5.4.2. The Eclipse Plug-in Architecture .......................................................................................... 59
   5.4.3. The TENCompetence Plug-in Architecture ....................................................................... 60
   5.4.4. TENCompetence Plug-in modules ....................................................................................... 61

   5.5. Coordination Service .................................................................................................................. 62
   5.5.1. Servlets for the Coordination Service .................................................................................. 65
   5.5.2. General considerations for the Coordination Service’s servlets ....................................... 65
5.5.3. UserAccess ........................................................................................................................................66
5.5.4. User ..................................................................................................................................................67
5.5.5. LN ...................................................................................................................................................67
5.5.6. CM ...................................................................................................................................................68
5.5.7. Competence .....................................................................................................................................68
5.5.8. Resource ..........................................................................................................................................69
5.5.9. Action ...............................................................................................................................................69
5.5.10. Goal ...............................................................................................................................................69
5.5.11. Role .................................................................................................................................................70
5.5.12. Permission ......................................................................................................................................70
5.5.13. References and links: ................................................................................................................87
5.6. What we reused ....................................................................................................................................70
5.6.1. Objectives of the evaluation process .........................................................................................87
5.6.2. Evaluation process .......................................................................................................................87
5.6.3. Selected and reused tools .............................................................................................................87
5.7. Why REST in TENCompetence? ......................................................................................................87
6. Deployment view .....................................................................................................................................77
7. Developer Guidelines ...........................................................................................................................79
7.1. SourceForge and Bugzilla ...............................................................................................................79
7.1.1. Background ....................................................................................................................................79
7.1.2. Bugzilla ..........................................................................................................................................79
7.1.3. Eclipse integration with Bugzilla provided by Mylar .................................................................87
7.1.4. References and Links: ..................................................................................................................87
7.2. Unit Testing and JUnit ....................................................................................................................87
7.2.1. Background ....................................................................................................................................87
7.2.2. Integration with Eclipse .................................................................................................................87
7.2.3. JUnit 3 or JUnit 4? .........................................................................................................................87
7.2.4. JUnit TENCompetence source folder structure and naming conventions ................................87
7.2.5. Automated and Manual Testing frameworks .............................................................................87
7.2.6. Guidance for writing tests ............................................................................................................87
7.3. The Use of a CVS Repository in the TENCompetence Project ..................................................87
7.3.1. Background ....................................................................................................................................87
7.3.2. CVS and SourceForge ..................................................................................................................87
7.3.3. Access rights and user permissions ............................................................................................87
7.3.4. Integration with Eclipse ................................................................................................................87
7.3.5. Repository details ........................................................................................................................87
7.3.6. Modules and folder structure ......................................................................................................87
7.3.7. JUnit folder structure ...................................................................................................................87
7.3.8. Other considerations when using folders ...................................................................................87
7.3.9. House Rules ..................................................................................................................................87
7.3.10. Component Owners ....................................................................................................................87
7.3.11. References and Links: ................................................................................................................87
7.4. Coding conventions and guidelines ..............................................................................................87
7.4.1. Naming conventions .....................................................................................................................87
7.4.2. Conventions for plug-ins .............................................................................................................87
7.4.3. User Interface Guidelines ............................................................................................................87
7.5. Runtime, Platforms, Tools and Libraries .......................................................................................87
8. References ..............................................................................................................................................93
9. Appendices ..........................................................................................................................................95
9.1. Appendix 1 - The TENCompetence Domain Model (Version 1.0) ..................................................96
1. The class diagrams ................................................................................................................................97
2. The vocabulary .......................................................................................................................................97
9.2 Appendix 2 – Initial Requirements Report (Version 2.2) .................................................................118
Introduction ............................................................................................................................................118
1. Definitions .......................................................................................................................... 118
  1.1. Acronyms .......................................................................................................................... 118
  1.2. Actors ............................................................................................................................... 118
  1.3. Main research activities .................................................................................................... 120
    1.3.1. Knowledge resources .............................................................................................. 120
    1.3.2. Units of Learning and Learning Activities ............................................................... 121
    1.3.3. Competence Development Programmes ................................................................... 121
    1.3.4. Learning Networks .................................................................................................... 121
  1.4. External Services ............................................................................................................ 121
    1.4.1. Portfolio service ........................................................................................................ 121
    1.4.2. Positioning service ................................................................................................... 122
    1.4.3. Navigation Service .................................................................................................. 122
    1.4.4. Notification service .................................................................................................. 122
  2. Functional requirements .................................................................................................... 123
    2.1. Explore a Learning Network ......................................................................................... 123
      2.1.1. Objective of the use case ...................................................................................... 123
      2.1.2. UML use case diagram ......................................................................................... 124
      2.1.3. Involved user-stereotypes .................................................................................... 124
      2.1.4. Pre-requisites ....................................................................................................... 124
      2.1.5. Narrative description .............................................................................................. 124
    2.2. Improve Proficiency Level ............................................................................................ 125
      2.2.1. Objective of the use case ...................................................................................... 125
      2.2.2. Involved user-stereotypes .................................................................................... 126
      2.2.3. Pre-requisites ....................................................................................................... 126
      2.2.4. Narrative description .............................................................................................. 126
    2.3. Keep up to date ............................................................................................................. 128
      2.3.1. Objective of the use case ...................................................................................... 128
      2.3.2. Involved user-stereotypes .................................................................................... 128
      2.3.3. Pre-requisites ....................................................................................................... 128
      2.3.4. Narrative description .............................................................................................. 129
    2.4. Reflect on competences ............................................................................................... 131
      2.4.1. Objective of the use case ...................................................................................... 131
      2.4.2. UML use case diagram ......................................................................................... 131
      2.4.3. Involved user-stereotypes .................................................................................... 131
      2.4.4. Pre-requisites ....................................................................................................... 131
    2.5. Study for a new function or a new job .......................................................................... 133
      2.5.1. Objective of the use case ...................................................................................... 133
      2.5.2. UML use case diagram ......................................................................................... 134
      2.5.3. Involved user-stereotypes .................................................................................... 134
      2.5.4. Pre-requisites ....................................................................................................... 134
    2.6. Want some support ....................................................................................................... 136
      2.6.1. Objective of the use case ...................................................................................... 136
      2.6.2. UML use case diagram ......................................................................................... 137
      2.6.3. Involved user-stereotypes .................................................................................... 137
      2.6.4. Pre-requisites ....................................................................................................... 137
      2.6.5. Narrative description .............................................................................................. 138
  9.3 Appendix 3 - The TENCompetence Client Scripting Monkey .......................................... 140
  1.1 Background ..................................................................................................................... 140
    1.1.1 Screenshot .................................................................................................................. 140
    1.1.2 Things it can do .......................................................................................................... 141
    1.1.3 Usage .......................................................................................................................... 141
    1.1.4 Writing a Script .......................................................................................................... 142
    1.1.5 Running Scripts ......................................................................................................... 142
    1.1.6 Example Scripts ......................................................................................................... 142
    1.1.7 DOMs .......................................................................................................................... 143
    1.1.8 Links ............................................................................................................................ 144
  9.4 Appendix 4 - The TENCompetence QA Component ....................................................... 145
    1.1 Background ..................................................................................................................... 145
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1.1</td>
<td>Application Area</td>
<td>145</td>
</tr>
<tr>
<td>1.1.2</td>
<td>Functionality and Usage</td>
<td>145</td>
</tr>
</tbody>
</table>
## Version history

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
<th>Editor(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>04-12-2006</td>
<td>First complete version.</td>
<td>Ruud Lemmers (LogicaCMG) and Phil Beauvoir (University of Bolton)</td>
</tr>
<tr>
<td>1.1</td>
<td>20-12-2006</td>
<td>Processed comments from internal project review.</td>
<td>Ruud Lemmers (LogicaCMG) and Phil Beauvoir (University of Bolton)</td>
</tr>
<tr>
<td>1.2</td>
<td>24-08-2007</td>
<td>Added appendices for Domain Model and Initial Requirements (which contain the use cases)</td>
<td>Ruud Lemmers (LogicaCMG)</td>
</tr>
</tbody>
</table>
1. Executive summary

This deliverable has two objectives. The primary and most important objective for this Architectural Design document is to communicate the decisions made concerning software components, their external properties and their relationships to one another, together forming the architecture of a system, to all involved parties. By viewing the system models in a number of different ways the structure and main concepts that inform the system can be more easily grasped.

The secondary objective is to provide developer guidelines for all those developers wishing to contribute to the integrated system. Use of these guidelines will make the integration of components/services from other work packages and external parties easier.

The architecture described has been used for the first official software release - the “Antelope” release. After the release of Antelope, the architecture will be redesigned to conform to a SOA approach. It was not possible to create a SOA from the outset, because there were not enough available requirements and integration of existing systems and components caused problems.

Using the concepts from [1], the architecture model provides the following viewpoints:

1. Use case view: shows the main process for the use case, “Study for new function or job”. This use case typifies three of the four primary use cases that will be implemented in the Antelope release and thus shows the most important process of the system. (Creation of the primary use cases, of which there are six was done by WP2 (Requirements & Analysis). This work has been incorporated as Appendix 2 – Initial Requirements Report (Version 2.2))
2. Logical view: gives the high-level system view from a user’s viewpoint, by providing descriptions, pictures and diagrams of the functionality.
3. Implementation view: shows the system’s main components from a developer’s point of view, by providing descriptions of frameworks, applied design patterns and descriptions of significant classes.
4. Deployment view: provides a network topology view of the system by showing the different TENCompetence nodes (machines) and their connections.

The Process view from [1] is not described in this document, because the performance and scalability aspects will be analysed when redesigning for the SOA approach.

The developer guidelines will also help developers to write code that is easier to integrate into the TENCompetence system. Guidelines are given for using Bugzilla (a bug tracking tool), Unit Testing, use of CVS (a system for version control of source code), coding conventions, developing user interfaces, and for development tools.
2. Introduction

2.1. Aims and Objectives
This deliverable has two objectives. The primary and most important objective for this Architectural Design document is to communicate the decisions made regarding software components, their external properties and their relationships to one another, together forming the architecture of a system, to all involved parties. By showing models of the system from a number of different viewpoints, one can more readily grasp the structure and main concepts that inform the system. The document should assist architects, designers and developers in understanding the technical parts of the system without having to examine quantities of source code, and it should also aid researchers, analysts, power users, pilot developers, and so on in understanding the functionality that the system has to offer. Communicating the architecture to a wider public will also promote discussions about what the system should do and how it should work.

The secondary objective is to provide developer guidelines for all developers who wish to contribute to the integrated system. Using these guidelines will make the integration of components/services from other work packages and external parties easier.

2.2. Road to this architecture
The architecture has been created by Work Package 3, which is responsible for the “Technical Design & Implementation of the Integrated System”. According to the project plan, the first 18-month cycle will be spent on creating a system by integrating a number of existing tools. Requirements are to be delivered by Work Package 2 (“Requirements & Analysis of the Integrated System”) at month 18. In practice, the lack of requirements (“What should the system exactly do?”) and the difficulty in integrating existing systems and components led to a number of problems. A few prototypes were created, but their interface was not satisfactory. The publication by WP2 of six primary use cases (shown in Appendix 2 – Initial Requirements Report (Version 2.2)) and the creation of wire frames (mock up screens) ultimately solved many of the problems. The time lost as a consequence has meant that the current architecture is not currently based on a SOA approach. Designing and realizing a SOA would result in late delivery to the pilots of the project.

The architecture described in this document reflects the architecture as used for the first official software release, the “Antelope” release. After the release of Antelope, the architecture will be redesigned to conform to a SOA approach.
2.3. Reading guide

Using the concepts from [1] and adding developer guidelines, the remainder of the document can be divided into five main sections:

1. **Use case view**: shows the main process for the use case “Study for new function or job”, which typifies three of the four primary use cases that will be implemented in the Antelope release and thus shows the most important process of the system. (Creation of the primary use cases, of which there are six, was done outside the Technical Design & Implementation work package.)

2. **Logical view**: gives the high-level system view from a user’s viewpoint, by providing descriptions, pictures and diagrams illustrating the functionality of the system. The most important chapters describe the wire frames and the client data model.

3. **Implementation view**: shows the system’s main components from a developer’s point of view, by providing descriptions of frameworks, applied design patterns and descriptions of significant classes.

4. **Deployment view**: provides a network topology view of the system, by showing the different TENCompetence nodes (machines) and their connections.

5. **Developer guidelines**: guidelines to help developers in writing code that is easier to integrate into the TENCompetence system. Guidelines are given for using Bugzilla (a bug tracking tool), unit testing, use of CVS (a system for version control of source code), coding conventions, developing user interfaces, and for development tools.

Additionally there are appendices that contain the domain model and the initial requirements, both created by WP2, and functional descriptions of the Scripting Monkey and the QA plug-ins.

2.4. Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Full description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOB</td>
<td>Binary Large Object</td>
</tr>
<tr>
<td>CDP</td>
<td>Competence Development Programme</td>
</tr>
<tr>
<td>CVS</td>
<td>Concurrent Versions System</td>
</tr>
<tr>
<td>FOAF</td>
<td>Friend Of A Friend</td>
</tr>
<tr>
<td>OSGi</td>
<td>Open Services Gateway initiative</td>
</tr>
<tr>
<td>PCDS</td>
<td>Personal Competence Development System</td>
</tr>
<tr>
<td>QA</td>
<td>Question &amp; Answering</td>
</tr>
<tr>
<td>SOA</td>
<td>Service Oriented Architecture</td>
</tr>
</tbody>
</table>
3. Use case view

3.1. Introduction

The main guide for the analysis and design of the TENC system have been the Domain Model (shown in Appendix 1 - The TENCompetence Domain Model (Version 1.0)) and the primary use cases (shown in Appendix 2 – Initial Requirements Report (Version 2.2)), both created by WP2 (“Requirements & Analysis”).

All primary use cases are connected to the 'goal' class in the Domain Model. In the final version a Learner should be able to use the system for attaining any of the following goals:

1. Keep up to date with an existing function or job.
2. Study for a new function or job or improve a current job level.
3. Reflect on current competences to determine which functions and jobs are within reach, or to assist in defining new learning goals.
4. Improve the proficiency level of a specific competence.
5. Receive support for a non-trivial learning problem.
6. Explore the possibilities in a new field (learning network) to help define new learning goals.

The focus of the Antelope release is on use cases 2 and 4. All other use cases have some support, but are also missing functionality (for instance, there is only simple support for positioning and navigation and there is no mechanism for competence assessment). Use cases 2 and 4 share many similarities as they both define a CDP using the Learner’s current competences. Use case 4 can be seen as a specialized case of use case 2. By describing use case 2 in the next sections, the most important functionality for the Antelope release – setting up a personal competence development plan – of the System is shown.
3.2. Use case: Study for new function / job

In brief, the Study for new function / job use case realises a personal competence development plan for the Learner and lets them perform the actions to achieve the required competences for a specific new function / job.

The main flow of events for the use case is as follows:

1. The Learner selects the option “Study for new function / job”.
2. The Learner selects an existing function / job to study for.
3. The Learner searches for CDPs to achieve the selected function / job.
4. The Learner selects the CDP that seems best for his personal situation.
5. The Learner subscribes to the actions from the CDP and performs those actions.

An important alternative flow is the scenario where the Learner does not find a fitting CDP. In this case the flow proceeds as follows after step 3:

4. The Learner requests the System to create a new CDP for his personal situation.
5. The System creates a “Pending CDP” request and forwards this to possible CDP Authors.
6. An Author creates a new CDP for the Learner, possibly using automatically generated CDPs from a Navigation service.
7. The Learner selects the new CDP.

Figure 1. Use case diagram for “Study for new function / job”
8. The Learner subscribes to the actions from the CDP and performs those actions.

The creation of new functions / jobs, competences and even learning networks can also be part of the flow of events, but are too detailed to describe here.
4. Logical view

4.1. Analysis class diagram: Study for new job

The Study for new job use case is used to create an Analysis class diagram, and to show the concepts using boundary classes (the user interface), controller classes (business logic) and entity classes (data storage). The resulting diagram can be seen as a translation from user concepts to system concepts. The analysis classes give a conceptual overview of the System’s parts and are consequently helpful to both developers and software architects.

For the main flow, the Learner uses the New Job / Function GUI screens to create a CDP for their goal. CDP Logic and Learning Network Logic control this process and create / retrieve / update data of entities CDP, Competence, Function / Job and Learning Network.

The other parts in the diagram are required for the alternative flow. CDP Logic notifies the Agent Logic of the request for a new CDP. Appropriate Authors are notified via the Agent UI screens of this request and they create new CDPs using the Create CDP UI screens. While creating CDPs, the Author can use information from
external Navigation Services. The *Navigation Interface* takes care of exchanging information with these external services.

### 4.1.1. Details of the boundary, controller and entity classes

The boundary and controller classes are described in section 4.3 TENCompetence Client Wire Frames, which uses wire frames to show what the system will look like and to describe how the functionality and logic work.

Wire frames were created for the Antelope release to design the interaction flow of the system and to understand the business logic needed by the system. The TENCompetence client reflects this, by having no strict separation between the presentation and business logic. As a result, the software differs from the structure in the analysis class diagram. The release after Antelope separates these layers, and the structure of the resulting system will therefore conform better to the structure of the analysis class diagram.

The entity classes require two sections for their description. The first section is the data model for the client data model, which is described in section 4.4 TENCompetence Client Data Model. The second section is the data model for the actual storage in the database, which is described in section 5.1 DB data model.
4.2. **Design Packages**

In order to present a high level overview of the system’s functionality, the System is shown in a Design Package diagram, below. Such a diagram shows the functionality broken down into logical groups:

![Design Package Diagram](image)

*Figure. 3 Design Package Diagram*
Table 1. Descriptions of Packages

<table>
<thead>
<tr>
<th>Package</th>
<th>Description and reference</th>
</tr>
</thead>
</table>
| Core Wizards, Views and Editors | The core screens, and their logic, of the TENCompetence client: wizards, views and editors for creating / reading / updating / deleting CDPs, Learning Networks, actions, support questions, assessments, etc, etc.  
  - Details from a technical point of view: 4.2.1 GUI & Business Logic package  
  - Details from a user point of view: 4.3 TENCompetence Client Wire Frames |
| Plug-ins                     | The plug-ins which can be added to the system to provide additional functionality. The shown plug-ins are examples of actual plug-ins for Antelope. Details: 5.4 Plug-in architecture of Eclipse |
| Event Model                  | Framework functionality for the use of events to notify observers. Details: 5.3 Event Model |
| Domain Model                 | The classes and interfaces used in the TENCompetence client for the entities of the domain model. Details: 4.4 TENCompetence Client Data Model |
| Conduits Framework           | Framework providing a means for communication between the client application and online services. Details: 5.2 Conduits Framework |
| Coordination client          | Functionality for sending changes in the domain model to the database model and for retrieving data from the database model. Details: 5.2.3 Co-ordination/Subscription conduits |
| Sharing                      | Publishes resources to external services. Details: 5.2.2 Sharing Conduits |
| DB Model                     | Functionality for persisting (normally to one or more databases) and retrieving data. Details: 5.1 DB data model |

4.2.1. GUI & Business Logic package

Containing the most functionality of all the design packages by far, this section provides additional information about the contents of the GUI & Business Logic package.

Focussing on the GUI & Business Logic package, its functionality can be divided into the following high level groups:

1. Learning Networks: create / read / update / delete functionality, functionality related to competence maps,....
2. Competences & Competence Development Programmes: create / read / update / delete functionality, functionality related to sharing and subscribing of CDPs,
3. Actions: create / read / update / delete functionality, functionality for subscribing to actions and for completing actions, …
4. Knowledge sharing: resources, quality of objects (rating and annotating resources, tutors, courses, action plans, …).
5. Units of Learning: assessments.
6. Common: collaboration (overview of participants and their online status, forum functionality), rich text editor and infrastructural issues (authorisation, authentication, encryption).

The groups in their turn can be detailed by determining the functions they perform. The following component diagram shows these functions:
Figure 4. GUI & Business Logic package/class diagram
Additional candidate functionality

When defining the new architecture, there will be more sources (besides the wire frames) for determining and defining additional functionality. The use cases from WP2, the pilots from WP4 and functionality which is currently out of scope for WP3 are all sources for additional functionality. The table below presents a preliminary list of additional candidates.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discovery</td>
<td>For finding learning networks.</td>
</tr>
<tr>
<td>Chat</td>
<td>For on-line chatting with friends or participants.</td>
</tr>
<tr>
<td>Resources</td>
<td>Standard and advanced searching, support for distributed repositories, ranking services, alerts, adding repositories, removing repositories, logical deletion of resources, …</td>
</tr>
<tr>
<td>Scheduler</td>
<td>For scheduling actions on someone’s calendar.</td>
</tr>
<tr>
<td>Policies</td>
<td>Advanced policy management for sharing / editing / copying resources.</td>
</tr>
<tr>
<td>Newsfeeds</td>
<td>Reading RSS and Atom feeds.</td>
</tr>
<tr>
<td>Flickr, del.icio.us, 43things</td>
<td>Reading from and publishing to these popular sites, using their own protocol.</td>
</tr>
<tr>
<td>Learning Design</td>
<td>Integrated support for IMS Learning Design: authoring and playing LD.</td>
</tr>
<tr>
<td>Portfolio</td>
<td>For recording evidence of achievements.</td>
</tr>
<tr>
<td>QTI</td>
<td>Support for the Question Test Interoperability standard: importing QTI, exporting QTI, GUI for creating and answering questionnaires, …</td>
</tr>
<tr>
<td>Find Tutor</td>
<td>Algorithm for finding (the most) suitable tutor for support.</td>
</tr>
<tr>
<td>View usage</td>
<td>Reporting on usage of Learning Networks, CDPs, actions, resources, …</td>
</tr>
<tr>
<td>Trails</td>
<td>Recording and using of trail information.</td>
</tr>
<tr>
<td>Annotate</td>
<td>Annotation of objects (persons, actions, …).</td>
</tr>
<tr>
<td>Search</td>
<td>Standard and advanced searching of competences, learning networks, …</td>
</tr>
</tbody>
</table>

Table 2. Preliminary list of candidates for additional functionality

From functions to SOA

When transforming the architecture to a SOA, after the release of Antelope, an important step is the definition of loosely coupled services. The list of methods from the component diagram can be used as input for the service list.

Analysis of the functionality for division in loosely coupled services reveals a focus on three types:

1. Functionality that updates persisted data.
2. Algorithms (e.g. calculating the current position of a learner).
3. Views showing information regarding one or multiple types of persisted items.
Types 1 and 3 can use the component diagram as their basis, type 2 is analyzed separately. The result is the first version of the service list. The final list is made by adding services for infrastructural services and by optimising for performance and generality.

The performance can be optimised by ensuring that communication with servers isn't too frequent (no "chatty" interfaces). By adding a small piece of logic in the client, it will be possible to replace multiple calls to small services by one call to a bigger service. Bundling services like this reduces the number of service calls and thus improves throughput. Generic services help to minimize the number of updates to the service interface and thus help to minimize changes to a service’s clients, as an interface change means changes to all that service’s clients.

4.3. TENCompetence Client Wire Frames

4.3.1. Introduction

The core requirements for the integrated system are defined in a use case model, non-functional requirements and in a domain model. The delivered requirements were on a high, abstract level and needed to be refined in work package 3 in order to implement the system. Implementing these high level requirements into a working system imposed a major risk. Without detailed requirements there was the danger of a lack of common understanding of them. By creating an interactive wire frame design of the client application for the integrated system this risk was mitigated.

An interactive wire frame design consists of a mock-up of the main application screens and the major interactions between these screens. The wire frames can be reviewed by all stakeholders such as system developers, users, researchers and can provide insight into the integrated system from the end user perspective. The process of creating these wire frames is much less involved then creating the actual application and the resulting wire frames can be modified relatively easy.

The wire frame design led to further refined requirements at an early stage of the development process that otherwise would have led to more changes during the development of the application. Additional benefits will come from the early involvement of the pilots to ensure that the system meets their needs. Having the wire frame designed ensures that an early start can be made creating the user manual in parallel with the development of the system.

The wire frames discussed in the following section demonstrate the first release of the TENCompetence system, “Antelope”. The wire frames have been produced with a tool called ‘GUI Design Studio’. GUI Design Studio comes with a free viewer. Readers are encouraged to install this viewer study the wire frame’s design. The viewer can be downloaded from http://www.carettaSoftware.com/ and the wire frames are available from http://dspace.ou.nl/handle/1820/795.

The main aim of the Integrated System is the support of individuals in lifelong competence development. This aim is conveyed by the name of the TENCompetence client software: the Personal Competence Development System (PCDS). The focus of the application is to provide the user with tools to develop their competences. A user
is always proficient in a competence on a certain level. These levels are called proficiency levels. Competences are either selected by the user directly, like playing guitar at a beginner level or indirectly by selecting a function/job which usually involves a number of competences. For example, the function “beginner guitar player” might, aside from the musical competences, also include stage performance competences and press contact competences. In order to develop these competences the user defines an action plan listing all actions required to attain the competences at a certain proficiency level. This action plan is the central concept of the application and the application aids the user in defining the optimum action plan.

Figure 5. TENCompetence client wire frame

Figure 5 shows the main application screen consisting of a collection of different visual components which are described in more detail in next sections.

4.3.2. Editors and Views
There are two different types of components in the client application:

- Editors. These components show the content of an item from the data model and when applicable allow the user to modify these items. Each type of item has its own editor. The first version of the PCDS has Editors for the following items: Action plans, activities and routes. Each item opens in its own editor; an item can never be shown in two editors at the same time. As with a text editor, editing an item does not immediately change the stored item. After making changes a user has the option to save the changes or to discard the changes. All editors share the same screen area (the editor pane), this implies that only one (if any) editor is active at any moment in time. When resizing the application all extra space is allocated to this editor pane.

- Views. These contain additional information about the items shown in the active editor. Once a view is made visible it remains visible; it is not connected to a
specific item but it often shows information connected to the current active editor. In contrast to an editor, changes to a view are completed instantaneously. Each view may have its own toolbar containing relevant options.

4.3.3. My Action Plans
This view lists all action plans defined by the user of the application. An action plan is a collection of actions that will lead to the development of one or more competences to a specified level. Each action plan is preceded by a checkbox indicating whether the user has completed the action plan.

![Action Plan View](image)

Figure 6. Action Plan View

The View’s toolbar contains options for creating a new action plan and for removing an action plan from the view. By clicking on the new action plan button the user invokes the new action plan wizard which upon completion adds a new action plan to the list shown in the view. When opening an action plan it is shown in its own editor in the editor pane. If there was already an editor containing this action plan, this editor is activated and brought to the foreground.

4.3.4. Action Plan Wizard
The action plan wizard offers the possibility to add an action plan to the list of action plans. An action plan consists of a goal, defined by either a competence or a function/job and a plan to achieve this goal consisting of a collection of actions. The first step of the wizard is to define a goal. The second step of this wizard is to add actions to the plan.

4.3.5. Defining the Goal of the Action Plan
There are three different options to define the goal of the action plan. The first option is to improve a specific competence. The wizard will guide the user through the process of selecting a competence from the list of available competences.
This list is composed of all competences that are defined for all known learning networks.

A learning network is discovered via a directory service which maintains a directory of all known learning networks. Alternatively, the user may decide to create a new competence because no suitable competence could be found. The application deliberately does not distinguish between authors and learners. Each and every user may act as learner and author at the same time and is encouraged to contribute to the learning network. After the user has selected a competence, the user is guided by the wizard to select the proficiency level for this competence. This proficiency level defines the target level of the competence which acts as the goal for this action plan. Again, the user may also add additional proficiency levels to the competence when needed. This way the user can contribute to the learning network.

The second option for defining a goal for the action plan allows the user to select a new function. This option is very similar to the selection of a competence and again the user can add new functions or jobs to the learning network if no suitable functions or jobs can be found. A function/job can be considered to be a wrapper for a collection of proficiency levels of competences.

Finally, the third option allows the user to keep track of a function/job. The wizard shows a list of functions/jobs mastered by the user. The user can select one or more of these functions/jobs and thereby will be notified via the agent when the user no longer meets one or more of the requirements for these functions/jobs. This could happen for instance when the definition of a function/job changes. The user may act upon these changes by updating his competences for these functions/jobs.
After defining a goal the user can add the actions that help the user in reaching the goal to the action plan.

### 4.3.6. New Action Plan

The second step of the wizard helps the user to determine which actions should be added to the action plan. These actions will help the user to develop the selected competence and acts as a personalized and planned competence development program.

![Figure 9. Define Action Plan Wizard](image)

There are three options to define an action plan. The first option allows the user to discover existing action plans that aim to develop the same proficiency levels. The second option allows the user to send a request to the learning network community for a new action plan. Other users may react to this request by creating this action plan. The request itself will be shown in the agent and will be visible to all other users in the learning network. Whenever an action plan is created for this request a message is sent to the agent to inform the requestor of this fact. Finally, the third option is to define the actions of the action plan from scratch and thereby create a new action plan.

### 4.3.7. Action Plan

The user can open an action plan from the view containing the action plans. The action plan is opened in a separate editor in the editor area. Each editor is represented by its own tab, in this case ‘Beginner guitar player’. This editor consists of three major areas that are described in detail in the following sections.
The action plan area shows the ordered collection of actions that have been added to achieve the goal for this action plan. Initially these actions are added via the action plan wizard. The actions presented are either activities or competence development programmes (CDP) that have the same objectives as the action plan itself. By performing these actions a user should reach the goal of the action plan.

For each action a status is maintained informing the user about the progress of this action. The progress indicator is represented by a checkbox indicating whether a user has started and/or whether a user has marked the action as completed. In the Antelope release, it is left up to the user to indicate if an action is completed or not. A user may also plan an action by adding a start and end date for it. In the Antelope release this feature is purely informative.

The actions can be either activities which act as leaves in the hierarchy or CDPs which represent a new level in the hierarchy. A CDP may contain activities and other CDPs. By double clicking on an action the action is opened. For an activity this means that a separate editor is opened showing the content of this activity. For a CDP this means going down in the hierarchy. The navigation bar at the top shows the current position within the hierarchy and can be used to move one level up the hierarchy.

A user may decide that the action plan is not adequate for achieving the goal. There may be additional actions which would complement the goal, or there may be actions the user considers obsolete. By using the Add and Remove buttons the user may customize the action plan.
Furthermore, the order of the actions, which represents the suggested order to perform the actions, may be modified by the user using the Up and Down buttons. When adding a new action to the actions list, by clicking the add button, the user is presented with a list of available actions, both activities and CDPs, from within the learning network. A user may select an action based on its rating or the user may decide to have a quick preview by double clicking the action. If no suitable action can be found, the user may decide to create a new action by selecting the ‘create a new action’ option. The user is guided by the wizard through the process of creating a new activity or CDP.

After changing the action plan, the user may decide to share this action plan with the learning network community. The action plan will be made available as a CDP leading to the goal and any user wanting to achieve the same goal may base the action plan on this newly created CDP. By sharing action plans in this way the user contributes alternatives to the learning network to achieve a specific goal.

The competence checklist area can be enlarged by clicking on the toggle button. A list of the competences that must be developed for achieving the goal of the action plan is presented.
The progress of the user with regards to these competences is represented by a checkbox. In the Antelope release the status is either acquired or not acquired. Besides the title of the competence a column is shown containing a planned action for this competence. This planned action is an action from the action plan that leads to the acquirement of the competence. This way the actions in the action plan are further refined giving insight into the purpose of each action with respect to the overall goal. An action can be associated with a competence from the competence checklist by selecting the ‘Plan Action’ button.

A wizard is launched showing all actions in the learning network.

This could be a long list that is not appropriate so this list can be limited by applying a filter. The list may be limited to all actions that are available in the current action plan or alternatively the list may be filtered on all actions that will develop the selected competence. If no suitable action is found, the user may decide to create a new action. This process is similar to creating a new action for the action plan as described earlier.

The user may also decide to modify the competence checklist. The reason may be that the user has a different understanding of the competence or function/job. The user may add or remove competences by using the Add and Remove buttons. This new competence checklist may be shared with other users in the learning network by
pressing the ‘Share your competence checklist with others’ option. The competence checklist will be added as a new competence or function/job to the competence map of the learning network. This way the user can again contribute to the learning network.

Finally, the description area allows the user to enter a description of the action plan. The user may do this to document the action plan for personal use. However, when the action plan is shared with the learning network this description will provide guidance for other users when using this action plan.

4.3.8. Activity Editor

Each activity can be opened in a separate editor. This editor presents all information of the activity to the user so the user can study and/or edit this activity.

![Activity Editor](image)

Figure 16. Activity Editor

Initially the activity is read-only meaning the user is not able to modify any of the contents of this activity. Furthermore the resources are not available to the user yet. By clicking the ‘participate’ option the user gains access to the activity and becomes part of the community belonging to this activity. The community is shown in the persons view, see Persons. Also the complete option becomes available enabling the user to mark the activity as completed. The editor consists of four major areas. The first area contains the description of this activity. The description instructs the user how to perform this activity. The description may be modified by all users. Changes made by one user are visible to all other users of this activity. So there is only one instance/version of this activity in the learning network which is shared by all users of the learning network.

The resource area contains the learning materials needed to successfully perform the activity. In the Antelope release, resources are either URLs or assessment resources. A user may add new resources to this list by clicking the add button or alternatively remove resources by clicking the remove button.
The objective area can be opened by clicking on the toggle bar. The objectives of this activity are represented by proficiency levels of competences that are achieved by this activity. Again the user may add or remove objectives by clicking the appropriate buttons. Using the properties button the user can view detailed information about the listed proficiency levels.

Finally the user can open the about area by clicking the toggle button.

The about area presents meta information about this activity like the original author of the activity, the type of the activity, the start and end time and the user license.

### 4.3.9. Competence Development Program Editor

![Figure 19. Competence Development Program Editor](image-url)
A Competence Development Program (CDP) consists of a collection of actions being either activities or CDPs. Furthermore, a CDP has objectives defining which proficiency levels will be developed by taking up this CDP. They are generated when a user shares an action plan. Via an edit option in the action plan a user can edit a CDP which will be opened in a separate editor. As is the case with activities a CDP is initially read-only. The user is not able to modify any of the content. By participating the user gains access and becomes a member of the community of this competence development program. Similar to the activity editor, this editor is divided into four different areas. First there is an ordered list with actions that will lead to the objectives of this CDP. These actions will appear in the action plan when a user defines a new action plan based on this CDP. The action list can be modified by the user and all modifications are global for the learning network. Again, this is similar to the activity editor. The objectives area contains a list of proficiency levels that are developed by this CDP. This list can be maintained by the user via the Add and the Remove buttons. Using the Add button a wizard allows the user to add either an existing proficiency level or to create a new one. Allowing a user to create the necessary proficiency levels in place enables the learning network to be created bottom up. Although the network still can be created using an overall design, using a bottom up approach makes extending and enhancing the learning network a much less involved experience and makes it feasible for more users to do so. The individual proficiency levels can be inspected and edited via the properties button. By providing this information the application can aid the user in selecting a proper CDP for obtaining a specific goal. The description area is used to provide information that guides the user who wants to use this CDP in an action plan. The “about” area provides meta information about this CDP.

4.3.10. Agent

The agent view presents personalized information to the user. This view is the visual front-end of the agent message service which enables different parts of the integrated system to communicate to the user. Communication is achieved by showing messages from the agent message service including hyperlinks providing shortcuts to functionality related to the message. Messages can be directed at the user specifically or at a community within a learning network. For example, when a user has an action plan monitoring a function or a job in order to keep up to date with this function/job, all relevant messages regarding this function/job are shown in the agent view for this user. The user may click the hyperlink to remediate the detected gaps in the competences for that function/job by adding an appropriate action plan.
4.3.11. Persons

The Persons view gives the user access to other people. These people are grouped into communities sharing the same interests. In the Antelope release two types of communities are supported. By clicking the appropriate button on the toolbar the user selects the community. The first community consists of the user’s friends based on the user’s FOAF file. The other community consists of the users having the same interests in the learning network. These interests are defined by either the goals of the action plan, the individual actions being activities or by the competence development programmes. Whenever an editor is open and active this view shows the community belonging to the content of the editor. For each user listed in the view a presence indicator is shown. The users can be contacted and the details of the user can be viewed. This information comes from the FOAF file of the selected user. This view also gives access to the user’s own FOAF information and enables the user to edit this information. Amongst other information, a picture can be uploaded that represents the user. Adding such an avatar gives the application a more personalized look and feel.

4.3.12. Rating

The rating view shows opinions of other users on a particular subject. The subject of the rating is the item visible in the active editor. When viewing an action plan this becomes the subject of the rating and when viewing a CDP or an activity in an editor those become the subject. The view gives the user an option to add a comment. A dialog is shown in which the user expresses an opinion by giving a rating on the
subject. This rating can be from zero to five stars where five stars is the highest rating and zero the lowest. Additionally, the user can add a comment on the given rating. The application calculates an average rating of the community on the subject and shows this community rating above the list of the individual ratings. The list shows all individual ratings given by the community on this particular subject. By clicking on a rating a dialog is presented showing this individual rating. This dialog shows the author, the rating, the comment made and the date the rating was posted. The user navigates to the other ratings using the Up and Down arrows in the top right corner of the dialog. This allows easy navigation without the need to close the dialog, select another rating and reopen the dialog again.

4.3.13. Support

![Figure 23. Support View](image)

The support view gives the user a place to request support or to discuss the current active subject. The active subject is defined by the open and active editor. For competence development programme editors and for activity editors this is the CDP or activity shown in the editor. For an action plan the goal of this action plan is the subject. By making the goal the subject of the support topic instead of making the action plan itself the subject, a community is formed around the goal instead of on a particular approach to that goal as expressed by an action plan. This way, discussing alternative approaches to achieving the goal is stimulated. Users can propose a different, and in their opinion, better action plan for developing the competences associated with this goal. This support view is implemented using a forum service.

4.4. TENCompetence Client Data Model

4.4.1. Introduction

This chapter describes the data model that is used by the TENCompetence client application. The client data model is implemented as a collection of Java objects that can be persisted. The data model presented in this chapter is the technical representation of the domain model from Appendix 1 - The TENCompetence Domain Model (Version 1.0) as it was defined by WP2. The client data model is intended to be used by the TENCompetence rich client application and by all plug-ins contributing to this client. For this purpose the data model can be extended via the plug-in mechanism provided by the Eclipse framework.
4.4.2. Extension point for Data model

Figure 24 depicts an implementation diagram of the TENCompetence client data model. The client data model is structured according to the Observer pattern. The core component of this model is the TenccDataModel. This component provides two major functions. First of all it defines an Eclipse extension point allowing plug-ins to register their implementations of the IDataModelFactory interface. Via this extension it is possible to create a new instance for one of the defined data model classes. Each plug-in can define its own data model classes. The only requirement is that the created data model object should implement the ITenccObject interface.

The TENCompetence rich client application also registers its own implementation of the IDataModelFactory via this Eclipse extension point.

The second function of the TenccDataModel component is to act as the subject in the Observer pattern. For this purpose objects implementing the IDataModelListener can register themselves in order to be informed of changes in the data model. Whenever such a change, such as creating, changing or removing an object in the data model occurs, an ITenccDataModelEvent is sent to all registered listeners. Informed by the ReST approach, three different implementations of ITenccDataModelEvent are available being ObjectAddedEvent, ObjectRemovedEvent and ObjectUpdatedEvent.

The persistence handler implements the IDataModelListener interface and acts as observer in the Observer pattern. The PersistenceHandler is responsible for the
persistence of any data model implementing the IPropertyOwner interface. The PersistenceHandler decouples the data model and the underlying server implementation for the persistence of the data, protecting the TENCompetence client from any side effects when the server implementation changes. The ITenccObject interface extends the IPropertyOwner interface. Because all data model objects have to implement the ITenccObject interface, they can all be persisted by the persistence handler. The IPropertyOwner defines methods needed for the marshalling and de-marshalling of content of the objects implementing this interface. The PersistenceHandler is responsible for the communication with the TENCompetence server to deal with the actual persistence of any data. The ITenccObject is the root element for all data model classes. A data model object is identified by a URI and a data model object can be restored by the PersistenceHandler module by passing the URI of an object to it.

4.4.3. Endpoints

A learning network could contain many items therefore it is not feasible for the client data model to maintain a local instance of all these items. In most situations this is also not required because the client represents the view of the user on the entire network so only items such as actions, action plans that are relevant to the user are presented. Because the client data model only contains a subset of the entire data set the usual approach of representing links between these items by maintaining references between the object instances representing them will not work. The TENCompetence data model is a network of highly interrelated objects. Objects maintain all different kinds of relations to each other. For example a Route is the parent for many actions, it can be used by different action plans and routes and it has objectives in the form of competences. These competences in turn might be the objectives of a number of other actions and so on.

A non-functional requirement imposed on the client data model is the late binding of items. Late binding means an item is loaded from the server only when it is actually needed. When loading an action plan its contained actions are not loaded immediately, only when the application needs access to such a child action is the item transparently fetched from the server. This approach has the advantage of keeping the local client data model as small as possible. Furthermore, loading of an action plan does not involve a large initial amount of network traffic which on slower network connections could lead to unacceptable performance of the application. A derived requirement on the client data model is that the relations between items are navigable both ways. Because the order in which items are loaded is undetermined, loading an item which is at the other end of a relation could be initiated from either end of a relation. For example, a competence might be loaded and then the user may want to see actions that have this item as an objective, or the other way around. The user is looking at a particular action and wants to know the objectives of this action.
To meet all these constraints the client data model splits the relation between two items in the two so called “endpoints” of this relation. Each endpoint acts as a stub to the relation and contains enough information to enable the application to retrieve the related item. There are two types of endpoints, single endpoints that maintain a relation to one item and multiple endpoints maintaining relations to more items. This enables the client data model to express the 1-1, 1-n, n-1 and n-n type of relations.

### 4.4.4. Data Model

The next section describes the TENCompetence data model in more detail. Figure 26 depicts the high level overview of the UML class model containing the main classes of the TENCompetence data model. All classes of the data model extend the TenccObject class which, by implementing the ITenccObject interface, enables persistence of instances of these classes as described earlier. Relations between classes with the stereotype «endpoint», should be implemented through the endpoint mechanism described in the previous section.

The diagram can coarsely be divided into two parts. The left hand side of the class diagram depicts the actions and the right hand side depicts the competences. The User acts as connecting class between the left and right hand side of the diagram.
Actions all extend the TenccAction class and for the Antelope release three distinct actions are defined: Activity, Route and LearningNetwork. The latter is the container and entry point for all other persistable classes in the diagram. All Actions may have children. An Item acts as child for a parent-child relationship with a TenccAction and therefore defines the appropriate endpoint relationship. Depending on the type of TenccAction different types of Item are allowed in the parent-child relation. A Route can only contain Activities and other Routes, a LearningNetwork only contains Routes and an Activity may only contain Resources. Enforcing these parent-child relationship rules are constraints on the model. The Resources linked to an Activity form the environment needed to perform this Activity. Resources also extend the Item model in order to enable this parent-child relation.

Each Item has a containment relation with LearningNetwork. TenccActions are defined globally and do not contain any personalized settings like the completion, start date and end date. The personalized information is stored in a Subscription and a TenccAction can have multiple Subscriptions, one for each user. For each type of TenccAction there is also a Subscription class extending Subscription. These are: RouteSubscription, ActivitySubscription and LearningNetworkSubscription. The correct matching of a particular Subscription class with the corresponding TenccAction class is an obvious constraint on the model.

A User can have multiple Subscriptions and the collection of Users subscribed to the same TenccAction act as community for that TenccAction. A Subscription may be active or inactive indicating that a user is or is not participating in the community. A user may become an inactive member if a TenccAction is no longer of interest to that user. However, a Subscription is never deleted as it contains parts of the portfolio for that user.

A LearningNetwork has a CompetenceMap associated with it. For the Antelope release this will be exactly one CompetenceMap. A CompetenceMap is a container for all instances of Competences and Roles (is equivalent to a function/job). A Competence is described by one or more ProficiencyLevels which discretely divides a competence into levels of mastering. Similar to a Competence, a Role is divided into one or more RoleLevels. A RoleLevel represents a certain level of a Role. A RoleLevel itself is described via a number of ProficiencyLevels which should be attained in order to qualify for this RoleLevel. A ProficiencyLevel has a relationship with a User. This relationship represents the ProficiencyLevels already attained by a user. Attaining a ProficiencyLevel in the Antelope release is considered to be a binary attribute. A ProficiencyLevel is either attained or not. A ProficiencyLevel also has an optional relationship with a TenccAction. This relationship represents the objectives of that TenccAction. Whenever a user has successfully completed a TenccAction, the user also has attained the ProficiencyLevels associated with the objectives of this TenccAction. A User has also a relation with a number of RoleLevels representing the targets or goals of that user.
Figure 27. UML class diagram action plan

Figure 27 depicts a more detailed class diagram of the action plan. An ActionPlan is the personal planning component available to the user to schedule TenccActions for a particular goal. An ActionPlan is always associated with a User and a User may have multiple ActionPlans. The goal of an ActionPlan is represented by a collection of ProficiencyLevels. These ProficiencyLevels are either provided by the ProficiencyLevels of a RoleLevel or more directly by the ProficiencyLevel of a Competence. This distinction is abstracted by the introduction of the IProficiencyLevelProvider interface in the class diagram. An ActionPlan is always based on a Route and an ActionPlan can be exported to a Route at any moment in time. Associated with each TenccAction in the ActionPlan is a PlannedAction. A PlannedAction maintains the relationship between a particular TenccAction and the ProficiencyLevels that are covered by this TenccAction. In other words, the PlannedAction describes how a particular TenccAction contributes to achieving the goal of the ActionPlan.

4.4.5. Implementation of the client Data Model

The client data model described above is implemented as part of the core of the Eclipse TENCompetence rich client. The approach taken was informed by the Eclipse API model. The public API is defined via interfaces and a factory method. These interfaces safeguard plug-ins against unwanted side effects whenever the actual implementation of these interfaces change. The data model factory provides access to the individual data model members. Data model listeners may subscribe themselves to the data model in order to be informed of any changes in the data model.

The data model may be extended by plug-ins whenever needed via a special Eclipse extension point. Via this extension point a new factory implementing the “org.tencompetence.datamodelfactory” interface can be declared. Besides providing the implementing class, the plug-in should provide a namespace for its data model classes. This namespace is used to avoid name clashes between the data model classes added by every plug-in. It is obvious that this name space should be unique throughout the plug-ins. It is recommended to start the name space with the root package name of the sources of the plug-in which should be unique between plug-ins anyway.

Although not recommended, it is possible for plug-ins to re-use the actual implementations for the data model interfaces. This can be especially useful for the more generic parts like the Endpoint and the TenccObject. However plug-in
implementers taking this approach should be aware that their code is not safeguarded against side effects by modifications on the data model implementation.

### 4.4.6. Data Dictionary

<table>
<thead>
<tr>
<th>Data Model Class or Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActionPlan</td>
<td>An ActionPlan represents an ordered collection of PlannedActions that a user has planned to perform in order to reach a goal. The goal targeted by the user is expressed by one or more ProficiencyLevels.</td>
</tr>
<tr>
<td>Activity</td>
<td>An Activity is a description of an action that a user has to perform (or has performed) to meet some objectives. These objectives are expressed by zero, one, or more ProficiencyLevels. An Activity contains an activity description to tell a user what actions to perform. It also contains a list of zero or more resources. A user must subscribe to an Activity in order to access it.</td>
</tr>
<tr>
<td>ActivitySubscription</td>
<td>An ActivitySubscription relates a user to an Activity. A user needs to subscribe to an Activity in order to access it. The ActivitySubscription maintains all personalized information of the user about this Activity like e.g. the completion status, the start time end the end time. All ActivitySubscriptions of an Activity form the community of this Activity.</td>
</tr>
<tr>
<td>BinaryResource</td>
<td>A BinaryResource is a specialized version of a Resource which acts as a container for all types of data persisted in the LearningNetwork itself. This includes binary files as well as text files.</td>
</tr>
<tr>
<td>Competence</td>
<td>A competence is defined as the ability (‘disposition’) of an actor to act effectively and efficiently upon the events in an ecological niche (an occupation, a hobby, a market, a sport, etc.). A Competence has one or more levels called ProficiencyLevels. A Competence is part of one CompetenceMap.</td>
</tr>
<tr>
<td>CompetenceMap</td>
<td>A CompetenceMap contains a structured way to represent competences, proficiency levels (ProficiencyLevel) and the functions/jobs (Role and RoleLevel). A CompetenceMap has an owner. A LearningNetwork has only one CompetenceMap.</td>
</tr>
<tr>
<td>IProficiencyLevelProvider</td>
<td>An IProficiencyLevelProvider interface provides access to the ProficiencyLevels of the object implementing this interface. The interface is implemented by ProficiencyLevel which returns itself and by RoleLevel returning a collection of ProficiencyLevels.</td>
</tr>
<tr>
<td>Data Model Class or Interface</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Item</td>
<td>An Item is an abstract class that acts as a child of zero, one, or more TenccActions. Because Items can have more than one parent the relations between parents and the Items form a network hierarchy instead of a tree hierarchy. Items can be Resources or TenccActions. All Items are part of an overall collection of Items of the LearningNetwork.</td>
</tr>
<tr>
<td>LearningNetwork</td>
<td>A learning network is an ensemble of users, learning resources (actions &amp; resources) and a competence map which are mutually connected through and supported by information and communication technologies to support lifelong competence development. A learning network is defined on a domain that represents a profession or a domain of knowledge.</td>
</tr>
<tr>
<td>LearningNetworkSubscription</td>
<td>A LearningNetworkSubscription relates a user to a LearningNetwork. A user needs to subscribe to a network in order to access it. The LearningNetworkSubscription contains all the personalized information about the user in this network, e.g. start time. All users having a subscription to the same LearningNetwork define the community of this network.</td>
</tr>
<tr>
<td>PlannedAction</td>
<td>A TenccAction is included in an ActionPlan in order to reach a certain goal. The PlannedAction links a particular TenccAction in an ActionPlan to this goal expressed as a collection of ProficiencyLevels. The PlannedActions define the reasoning between an ActionPlan and its TenccActions and this reasoning is shared between the users sharing the same ActionPlan.</td>
</tr>
<tr>
<td>ProficiencyLevel</td>
<td>A competence has 1 or more proficiency levels that are an integral part of the competence itself. The proficiency level can be any range of Integer numbers and indicates the minimum level that needs to be attained for a certain RoleLevel in the domain. A user has zero or more ProficiencyLevels attained, thus defining the position of that user on the CompetenceMap. TenccActions also have relations with zero or more ProficiencyLevels defining the objectives of that action. Completing a TenccAction implies attaining the associated objectives/ProficiencyLevels. A collection of ProficiencyLevels also form the goal of an ActionPlan. Via PlannedActions individual TenccActions are associated with (part of) this goal.</td>
</tr>
<tr>
<td>Data Model Class or Interface</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Resource</td>
<td>Knowledge resources are any kind of resources that can be used in learning. Two types of Resources are defined in the Antelope release of PCDS. They are WebResources, providing pointers to resources on the web and BinaryResources, providing actual files as a resource. In addition to the actual resource self a Resource also maintains metadata about this resource like the title and a description of the resource. All Resources are linked to the LearningNetwork in which they are defined and they can be linked to as many Activities as is applicable. By adding them to an Activity the resources are placed in a learning context in which a user can use them in order to achieve a certain goal.</td>
</tr>
<tr>
<td>Role</td>
<td>A domain of knowledge can contain different actors with different Roles (in work called job functions/job titles). A Role is defined by the different Competences that a user needs to attain in order to qualify for the Role. A Role consists of at least one level called a RoleLevel. A RoleLevel defines the target-proficiency levels of the competences the user should obtain in order to qualify for a particular level.</td>
</tr>
<tr>
<td>RoleLevel</td>
<td>Roles can be performed at one or more role levels, like Trainer, Master and Trainee. A RoleLevel is defined by the ProficiencyLevels the user should attain in order to meet the criteria for this specific level of a Role. A RoleLevel can be the focus of learning, in other words can be the goal of an ActionPlan.</td>
</tr>
<tr>
<td>Route</td>
<td>A Route (synonyms: Competence Development Programme (CDP), learning path, curriculum, programme) contains an ordered set of activities and other Routes that have to be (or are) followed to attain certain objectives expressed as a collection of zero or more ProficiencyLevels. A user needs to subscribe to a Route in order to access it. This RouteSubscription contains personalized information from the user about this Route.</td>
</tr>
<tr>
<td>RouteSubscription</td>
<td>A RouteSubscription contains the personalized information from a user about a particular Route, e.g. the planned start and end dates, the completion status etc. All RouteSubscriptions of a single Route define the community of users of this Route.</td>
</tr>
<tr>
<td>Data Model Class or Interface</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Subscription</td>
<td>A Subscription is an abstract class containing personalized information about a user and a specific TenccAction. Examples of this type of personalized information are the planned start date of an action in an ActionPlan, whether the user has completed the action or not. A user may only access an action by subscribing to it. The act of subscribing instantiates a Subscription object for this user. A Subscription can be active or inactive for a particular user. An active Subscription indicates a user is participating in the community and an inactive Subscription indicates a user not participating. This active-inactive state may be switched by the user at will. Inactivating a Subscription will not remove it, because the personalized information for the user needs to be pertained.</td>
</tr>
</tbody>
</table>

The LearningNetwork consists of communities around topics in this network. For example, there is the encompassing community of all users in the network, but there are also communities around individual Activities and Routes. These communities are defined by the collection of subscriptions on the same TenccAction.

Subscription is an abstract class, so no instances of it can be created. A Subscription has three non abstract subclasses, ActivitySubscription, LearningNetworkSubscription and RouteSubscription that differ in the types of TenccActions they subscribe to and consequently in the different personalized information that is stored about these types of actions.
<table>
<thead>
<tr>
<th>Data Model Class or Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TenccAction</td>
<td>A TenccAction defines one learning object. This learning object can be an Activity, a Route or a Learning Network. Successfully completing a TenccAction leads to achieving its ProficiencyLevels. A user can only access a TenccAction after subscribing to it. Subscribing enables the application to keep track of the user’s personalized information on this action. All users subscribed to a TenccAction form the community of that action. TenccActions can have zero or more items as child elements. Depending on the actual type of the TenccAction (LearningNetwork, Activity or Route) different types of items are allowed. An Activity may only contain Resources; a Route contains only Activities and other Routes; and a LearningNetwork only contains Routes.</td>
</tr>
<tr>
<td>TenccObject</td>
<td>The abstract class TenccObject is the root class of the client data model. TenccObject defines all common behaviour of the classes from the data model. The two most prominent common features are support for the persistence of the data on the server and support for endpoint relations between data objects in the model. The latter is elaborated in more detail in the section about endpoints. All objects extending TenccObjects are persistable via the persistence handler.</td>
</tr>
<tr>
<td>User</td>
<td>A User is a person using the application. Use of the application may vary on the role a particular user has in the learning network. For example, a user acting as a learner wants to attain a certain set of competences via an action plan. An author would contribute learning materials to the learning network or refine the competence map associated with the network. A major aim of the application is to let each and every user take up any role or combination of roles. In fact the application even stimulates learners to act as authors in order to enrich the learning network with their contributions. By associating users with the ProficiencyLevels and RoleLevels they have acquired they are placed on the CompetenceMap. A user defines goals by creating an ActionPlan for each of these goals. The application uses the position on the competence map to help the user to navigate to these goals. In order to access a LearningNetwork, a Route or an Activity a user needs to subscribe to such a TenccAction. By subscribing the user becomes part of the community formed around this action.</td>
</tr>
<tr>
<td>Data Model Class or Interface</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>WebResource</td>
<td>A WebResource is a specialized type of Resource that acts as a link to a resource on the web. A WebResource contains a URI to this item and associates a title and a description with this resource.</td>
</tr>
</tbody>
</table>

Table 3. Descriptions of Data Model classes/interfaces
5. Implementation view

5.1. DB data model

5.1.1. Introduction
The TENCompetence domain model describes, among other things, a set of entities and their relationships as used within the TENCompetence system. A consequent requirement is that the TENCompetence System needs to store these entities. As already described elsewhere, the Coordination Service is the interface between the TENCompetence Client and the remote data, meaning that it is its responsibility to interact with physical information.

From a general architectural point-of-view the coordination service should be able to retrieve information from an external system or component. For example, resources should be located by calling WP5 services. For the purposes of the current release we need to have a storage system that is able to store all the data, until the other components can expose their services.

The first issue concerns the mapping of the domain model’s entities and relationships to a data model respecting its needs. There exists, where possible, a direct mapping from the domain model to the data model. This means that there should be a data structure for each entity; one for a Learning Network, one for a Competence and so on. The same applies for relationships.

The data model should also provide some additional data structure to support the client’s operation and requirements such as “Permission” and “Users”.

The data from the coordination service’s client are constantly evolving. This means that it is important to build a data structure which can support the storage of additional data. As the data from the client are XML documents the natural way to fulfil this requirement is to store all the additional xml data as they are received. This could be useful, for example, to store metadata. Not only important data could be stored in this way, in fact, all the data that is the result of a query could be described and accessible. If a change is made in important data a change is necessary also in the data structure.

5.1.2. Database Description
After an analysis of various options the DBMS has been chosen. The first decision was to adopt either a Relational DBMS (RDBMS) or an XML-based one (XDBMS). We considered the second option because the data exchanged between the TENCompetence Client and the coordination service are XML documents. We rejected the XDBMS option because they are not widespread yet and there would be no obvious advantages from this choice since in the future we need to retrieve information from many different sources. Therefore, we have chosen a classic RDBMS, MySQL.

The following schema describes the actual structure of the database.
Figure 28. Schema showing the structure of the database
The following database tables are derived from the domain model:

1. competence
2. competence_map
3. action
4. learning network
5. role
6. goal

Other database tables such as resource_action, action_user, resource_permission, user_competence are used to store the relationships between database tables. They also contain other important information. user_competence, for example, contains the proficiency levels reached by the users.

The most important database tables have a BLOB field named “xml”. The reason for this is that the definition of data exchanged is not yet fixed and may well change frequently, so it is preferable to store the data on which no queries are performed in a BLOB field. Moreover, we have defined specific fields for data we access frequently.

For the “action” table we need to also store information such as the option to add a new member. This is information we will access frequently.
5.1.3. Descriptions of tables

The following table describes the tables of the database:

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tables deriving from domain model</strong></td>
<td></td>
</tr>
<tr>
<td>Action</td>
<td>Contains Action data (to store Activity and Route data, which are specialized versions of an action). In particular store information such as allowNewMember.</td>
</tr>
<tr>
<td>User</td>
<td>Contains Users registered in the system.</td>
</tr>
<tr>
<td>Competence</td>
<td>Contains the Competences available for Users. An important field is competence_map_id which connects a Competence with a Competence Map.</td>
</tr>
<tr>
<td>competence_map</td>
<td>Contains the Competence Map.</td>
</tr>
<tr>
<td>Goal</td>
<td>Contains the Goals of the User.</td>
</tr>
<tr>
<td>learning_network</td>
<td>Contains the Learning Network.</td>
</tr>
<tr>
<td>Resource</td>
<td>Contains the Resources description of the system.</td>
</tr>
<tr>
<td>Role</td>
<td>Contains the Roles available in the system. competence_map_id connects a Role (function/role) with a Competence Map.</td>
</tr>
<tr>
<td><strong>Relation Tables</strong></td>
<td></td>
</tr>
<tr>
<td>action_learning_network</td>
<td>Used to store relation data between an Action and a Learning Network.</td>
</tr>
<tr>
<td>action_user</td>
<td>Contains the relation between Action and User. In particular describes the ownership of an Action.</td>
</tr>
<tr>
<td>learning_network_user</td>
<td>Used to store the users subscribed to a Learning Network.</td>
</tr>
<tr>
<td>permission_user</td>
<td>Contains the permissions that a User has for a Resource.</td>
</tr>
<tr>
<td>resource_action</td>
<td>Contains relations between Resources and Actions.</td>
</tr>
<tr>
<td>user_resource</td>
<td>Contains User Assessments for Resources. completion_status: text that describe the Resource completion status for one User preference: double that represents the User preference for the Resource</td>
</tr>
<tr>
<td>user_competence</td>
<td>Contains relations between Users and Competences level: string that describes Competence Level for the User</td>
</tr>
<tr>
<td>user_role</td>
<td>Contains relations between Users and Roles.</td>
</tr>
<tr>
<td><strong>Other Tables</strong></td>
<td></td>
</tr>
<tr>
<td>Code</td>
<td>Contains the permission codes that a User can have for a Resource. A specific table is used to allow the definition of new permission codes without any modification to the code.</td>
</tr>
</tbody>
</table>

Table 4. Descriptions of Database Tables
5.2. Conduits Framework

5.2.1. Description
The Conduits Framework is an Eclipse-based plug-in framework for the TENCompetence Client that provides a means for communication between the client application and online services. It is derived from the “Plex” Personal Learning Environment application prototype developed at Bolton University.

The Conduits Framework currently provides two types of “conduit”:

1. **Sharing conduits** such as publishing a description of a Resource(s) to service APIs such as Blogger (www.blogger.com) or Simpy (www.simpy.com), and OPML (www.opml.org) and Atom-based (http://en.wikipedia.org/wiki/Atom_(standard)) services via their APIs. Publishing of details of Resources in this way allows a user to share the Resources’ details with other interested users by using publicly-available services rather than by, say, a direct peer-peer connection.

2. **Co-ordination/Subscription conduits** intended to provide a means of synchronisation of the Data Model between client and the server-based services such as Learning Networks.

The original Conduits Framework also supports query-based protocols, such as searching the 43Things service via its API or querying a XCRI-based service, though at present this type of conduit has not been ported over from Plex to the TENCompetence code base.

It is an extensible framework that utilises the Eclipse Extension Point mechanism enabling developers to write their own protocol extensions.

5.2.2. Sharing Conduits

Usage in the TENCompetence Client
The Sharing Conduits implementation appears in the TENCompetence Client in two places:

1. A Conduits Manager dialog box that enables the user to set up and configure Conduits.

2. A contextual menu item (“Share…””) that is enabled when a user selects an appropriate Resource. Selection of this menu item invokes a dialog box allowing the user to select the chosen Conduit from a selection of previously configured Conduits, add optional comments and tag words, and then publish the Resources’ details together with the tags and comments to the target service location.
An example of use would be where a user accesses the Resources View in the TENCompetence Client and selects a given Resource. A right-click menu item selects the “Share…” menu item. In the resulting “Share Resource” dialog box, the user selects their “Blogger” conduit (which has been previously configured), and adds some comments and relevant tags. Clicking on the “Publish” button transmits this information to the receiving Blogger service and a corresponding new entry added to the user’s Blogger page.

**Sharing Conduits, Protocols and Configurations**

A Sharing “Conduit” is really a connection protocol that conforms to a given service API. Conduits are configured to create User Conduits in the Conduits Manager dialog box available from the “Tools” menu in the TENCompetence client. Thus what the user sees as a Conduit is really a specific configuration of an implemented communication protocol.

Supported connection protocols that have been implemented to date are as follows:

- Atom Publishing Protocol (used by Blogger)
- Del.icio.us API
- OPML Publishing Protocol
- Simpy Social Bookmark API

To configure a Conduit that connects to the public API provided by Blogger (http://www.blogger.com), the information required to create a User Conduit configuration is as follows:

- **Conduit Name:** “My Blogger Space” (or whatever)
- **Protocol:** Atom Publishing Protocol
- **Location:** URL (example - https://www.blogger.com/atom/19327500)
- **User Name:** User name for the Blogger account
- **Password:** User’s password for the Blogger account

This “My Blogger Space” Conduit configuration will then appear as an available Conduit from the “Share Resource” dialog box.

Container objects such as Folders and future implementations such as Activities may also be shared, or published by Conduits that implement the necessary interfaces of IFolderConduit and IActivityConduit. It is up to the implementing Conduit class to define what information is transmitted to the receiving service. In the case of a Folder container, the child elements are usually transmitted as a series of multiple Resources, whilst in the case of an Activity, the Activity description and associated metadata may be transmitted.
The public API interfaces for the sharing conduits are in the package `org.tencompetence.tencc.conduits`.

The main interface is defined as `ITenccConduit`. Sub-interfaces define further functionality needed for certain types of Conduit. The main (abstract) Conduit class is defined as `AbstractConduit`.

- `ITenccConduit` - defines the main methods necessary for a Conduit.
- `ITenccSharingConduit` - at present defines no methods instead acting as a marker/placeholder interface. However, it will possibly contain defining methods in the future. Further sub-interfaces can also be defined in the future as required.

**Figure 29. Class and interface structure of the Sharing Conduits Model**
IFolderConduit - defines the methods necessary for sharing a folder container object.

IResourceConduit - defines the methods necessary for sharing a resource object.

IACTivityConduit - defines main methods necessary for sharing an Activity object in a future implementation.

AbstractConduit - implements the methods of ITenccConduit.

Concrete implementations are provided in the “impl” sub-package. These are presently:

- OPML
- Simpy
- Atom
- DelIcioso

The Conduit Configuration framework is provided in order to allow the creation, editing and persistence of configurations of Conduits. The Blogger example, above, is a configuration of the Atom Conduit. Configurations are persisted locally and retrieved by the ConduitManager class.

IConduitConfiguration - defines the methods necessary for a single configuration of a Conduit.

ITaggedConduitConfiguration - defines further methods necessary for setting and getting tags required for Conduits that support tagging when publishing.

AbstractConduitConfiguration - implements the methods of IConduitConfiguration

TaggableConduitConfiguration – Concrete implementation of ITaggedConduitConfiguration and AbstractConduitConfiguration.

ConduitDescriptor - A lightweight descriptor class created by the ConduitManager as it retrieves the declared conduit extensions from the plug-in manifest. It describes an ITenccConduit, so that one can get information about the ITenccConduit without having to instantiate it until required.

ConduitManager – The main Conduits Manager class that manages the loading and saving of Conduit Configurations and Descriptors.
Extension point for developers

The Sharing Conduits Framework defines an Eclipse extension point named
org.tencompetence.tencc.conduits.sharing_conduits. Developers who wish
to extend this in order to provide further communication protocols may do so by
declaring an appropriate extension point that implements an instance of
org.tencompetence.tencc.conduits.ITenccSharingConduit. As noted above,
current implementations are:

- Atom - org.tencompetence.tencc.conduits.impl.Atom
- Del.icio.us - org.tencompetence.tencc.conduits.impl.DelicioUs
- OPML - org.tencompetence.tencc.conduits.impl.OPML
- Simpy - org.tencompetence.tencc.conduits.impl.Simpy

Each of these classes implements
org.tencompetence.tencc.conduits.ITenccSharingConduit and extends
org.tencompetence.tencc.conduits.AbstractConduit.

By providing further extensions to this point, the implemented protocol(s) will
automatically appear in the drop-down list of selectable protocols in the Conduits
Manager dialog window.

5.2.3. Co-ordination/Subscription conduits

Usage in the TENCompetence Client

The co-ordination Conduits implementation appears in the TENCompetence Client in
two places:

1. A main menu item (“Join Network…”) that is by default enabled. Selection of
   this menu item invokes a dialog box allowing the user to specify a URL of a given
   subscription service. Further subscription options are displayed allowing the user
   to subscribe to a Learning Network or Activity Service.
2. A contextual menu item (“Refresh…”) that is enabled when a user selects an
   appropriate object in the client. Selection of this menu item invokes actions that
   remain unimplemented but are planned as a synchronisation between client and
   server object.
Class structure

The public API interfaces for these conduits are in the package
org.tencompetence.tencc.coordination.

The main interface is defined in this package as ICoordinationConduit which extends org.tencompetence.tencc.conduits.ITenccConduit. Sub-interfaces define further functionality needed for certain types of co-ordination Conduit. The main functionality is contained the class CoordinationManager.

CoordinationManager – The main class that implements org.tencompetence.tencc.datamodel.IDataModelProvider handling data model events synchronising these to the current network subscription. This class is based on org.tencompetence.tencc.conduits.ConduitManager and shares the same saving and loading functionality of conduit configurations and descriptors.

ICoordinationConduit – defines the main methods necessary for a Co-ordination Conduit.

ICoordinationFeed – defines the main methods necessary for a Co-ordination Feed Conduit.

ICoordinationProvider – extends ICoordinationConduit and ICoordinationFeed. Defines no additional methods as yet.
TenCompetenceSubscription – Concrete implementation of org.tencompetence.tencc.conduits.AbstractConduit and ICoordinationProvider.

**Extension point for developers**

The Co-ordination Conduits Framework defines an Eclipse extension point named org.tencompetence.tencc.coordination.providers. Developers who wish to extend this in order to implement further data model providers may do so by declaring an appropriate extension point that implements an instance of org.tencompetence.tencc.datamodel.IDataModelProvider. As noted above, the current implementation is the CoordinationManager.

### 5.3. Event Model

The TENCompetence client for the Antelope release is designed to support an extensible architecture using the plug-in concepts provided by the Eclipse framework. Plug-ins may extend the client application with new data model elements, interface components such as views and editors or with additional conduits providing access to services. These plug-ins must cooperate with each other to maintain consistency through the application. However, it is impossible to determine beforehand which other objects should be informed about changes in an object. Hence a tight coupling between objects is not desirable and even impossible in the case of the TENCompetence client. A solution to this problem has been described in the Observer pattern ([2]). The general idea behind this pattern is that “…when an object of the data model (the Subject) is changed, it generates an event. Other objects can subscribe to these events: the Observers. When an event is generated, it is sent to its observers.” In this way, the subjects don’t need to know anything of their observers, thus ensuring a loose coupling between objects and components.

Figure 31 shows a UML class diagram depicting the implementation of the Observer pattern in the TENCompetence Client application.

![UML class diagram of the TENCompetence Client Event Model](image)

The TenccDataModel implements the ITenccDataModel interface which contains all methods that are relevant for the subject in the Observer pattern. There is only one instance of the TenccDataModel which is guaranteed by following the Singleton pattern ([2]). The abstract class Observer represents an observer in the Observer
pattern. Many classes in the TENCompetence client, such as the PersistenceHandler and Views act as observers. All these observers must implement the IDataModelListener interface and are responsible for registering and de-registering themselves as listeners with the subject, which is the singleton class TenccDataModel. It is important to notice that the subject does not need to be aware of any components that wish to be informed of changes in the data model. The onus of responsibility is now moved to the component that wishes to be informed of changes in the data model.

The TenccDataModel notifies registered observers with three types of events: ObjectAddedEvent, ObjectRemovedEvent and ObjectUpdatedEvent. The events are fired when an ITenccObject is created, deleted or changed. The implementations of ITenccObject are responsible for notifying the TenccDataModel about the occurrence of changes in the object. When creating the event this object adds itself as payload to the event, thereby giving the observer access to the object triggering the event. The TenccDataModel provides a transaction mechanism. This mechanism aggregates incoming events into a single transaction. The rationale behind this transaction mechanism is to avoid unnecessary load on the system by sending out too many notifications to interested observers. This would not cause a problem for components that have little overhead like GUI components, however components such as the PersistenceHandler have a large communication overhead with the TENCompetence server and should be notified as little as possible.

Each transaction should start with a call to startTransactions() and end with a call to commitTransactions(). These calls are typically called by GUI components which react to user interactions that change the data model. Figure 32 depicts a UML sequence diagram showing an example of the TENCompetence event model. The figure shows an Observer registering itself to the TenccDataModel in order to be notified about changes in the data model. Next, a user creates a new element via a NewElementWizard. The sequence diagram picks up when the user finishes the wizard by pressing the finish button.

![UML Sequence Diagram](image)

**Figure 32. UML Sequence Diagram of the TENCompetence Client Event Model**

The NewElementWizard starts a new transaction in step 3. Next, a new ITenccObject is created in step 4, and in step 5 the TenccDataModel is informed about the creation
of the new object by adding an instance of the ObjectAddedEvent to the transactions. In step 6 the title of the newly created object is changed and the TenccDataModel is again informed about this change in step 7 via an instance of the ObjectUpdatedEvent. The transaction is closed by the NewElementWizard in step 8 via a call to commitTransactions. The TenccDataModel now starts optimizing the collection of events. Optimizing the events collection consists of removing unnecessary events which have become obsolete by other events like for example clearing all events for an object which occurred before a remove event of that same object. Next, all attached observers are notified of the transactions in step 9. The optimized collection of events is passed to the observers as an argument of the dataModelEvents call.

5.4. Plug-in architecture of Eclipse

5.4.1. Background
Plug-ins embody an architectural pattern for building an application from constituent parts. It is both convenient and advantageous to follow this pattern in order that developers may contribute discrete components at any stage of the development process. The key elements provided by this pattern are:

- Modular component structure
- Discrete development process
- Extensibility
- A convenient updating process
- Public API

This pattern is to be adopted by the TENCompetence project in its development of its client technology. Eclipse is the target platform for the TENCompetence Client and supports a powerful plug-in architecture.

5.4.2. The Eclipse Plug-in Architecture
Eclipse is an extensible platform that provides a core of services that work together to support various tasks. Developers contribute to the platform by delivering their tools in pluggable components, known as plug-ins, which conform to Eclipse's plug-in contract. The basic mechanism of extensibility in Eclipse is that new plug-ins can add new processing elements to existing plug-ins.

The net result is interoperability of components, developed discretely so that they co-exist without conflict. Furthermore, contributed components may themselves be extended so that a plug-in may build on top of another plug-in. Tied in with the concept of plug-ins is Eclipse’s component architecture consisting of the following:

- Runtime – provides the plug-in infrastructure, loading and OSGi framework ([4])
- Standard Widget Toolkit (SWT) – native graphics and widgets
- JFace – A set of UI MVC (Model-View-Controller) frameworks built on top of SWT
- Workbench - UI layout of windows, views and editors and perspectives
Plug-ins can contribute to and extend any of the above component levels.

Every component in the Eclipse framework is a contribution. Each contribution is packaged as a plug-in that is only loaded by the system as and when needed, since many plug-ins could be potentially loaded ultimately slowing down the system. Thus, plug-ins are only loaded when they are first called upon. This is achieved by declaring the details of the plug-in in an XML-based manifest file. The manifest file describes the contents and capabilities of the plug-in. The net result is a system that loads quickly and remains lightweight and agile.

5.4.3. The TENCompetence Plug-in Architecture

The TENCompetence software follows this Eclipse based plug-in pattern. In the case of the TENCompetence Client, there is a core component (which itself is a plug-in) which acts as the host (org.tencompetence.tencc). This core plug-in should be supplemented by auxiliary plug-ins that contribute additional functionality to the host. For example, the Question and Authoring component is provided as the org.tencompetence.tencc.questionauthoring plug-in.

It is important to stress the importance of the plug-in architecture for the TENCompetence development process. No piece of functionality should be seen as too trivial to reside in its own plug-in, as long as it offers a unique component contribution. Developers should be encouraged to adopt this paradigm at all times. An additional advantage is that components can be easily updated and maintained by means of the Eclipse OSGi/Equinox framework.
5.4.4. TENCompetence Plug-in modules

The following is a list and brief description of the existing plug-ins in the TENCompetence software framework to date:

<table>
<thead>
<tr>
<th>Plug-in module name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.tencompetence.tencc</td>
<td>The main Client Application Host containing windows, dialogs, runtime components.</td>
</tr>
<tr>
<td>org.tencompetence.tencc.conduits</td>
<td>Sharing and Provider Conduits Framework</td>
</tr>
<tr>
<td>org.tencompetence.tencc.conduits.coordination</td>
<td>Co-ordination and Subscription Conduits Framework</td>
</tr>
<tr>
<td>org.tencompetence.tencc.crypto</td>
<td>Cryptographic storage utilities</td>
</tr>
<tr>
<td>org.tencompetence.tencc.gui</td>
<td>GUI utilities</td>
</tr>
<tr>
<td>org.tencompetence.tencc.monkey</td>
<td>Scripting Monkey Module</td>
</tr>
<tr>
<td>org.tencompetence.tencc.questionauthoring</td>
<td>Question Authoring Module</td>
</tr>
<tr>
<td>org.tencompetence.tencc.server</td>
<td>Test Server</td>
</tr>
<tr>
<td>org.tencompetence.tencc.tencs</td>
<td>Coordination service of the database server.</td>
</tr>
</tbody>
</table>

Table 5. Existing plug-ins in the TENCompetence software framework to date

More modules are planned as development proceeds.
5.5. Coordination Service

The TENCompetence System needs to access a number of mixed resources such as media files, user information, and competency data. The system thus requires integration of a number of heterogeneous repositories that can provide access to these resources. This task is assigned to the Coordination Service. The Coordination Service is the entry point for accessing resources. This Service also exposes an interface to external systems.

The following figure depicts the main functionality of the component:

![Diagram](image)

**Figure 33. Main functionality of the Co-ordination Service**

So far, we have completed the definition of the overall architecture, the interface definition and the implementation of a first release internal repository. This repository is able to store the information and resources required by the domain model such as resources, users, learning networks, roles, competencies and so on.
The following diagram depicts the structure of the Coordination Service component for the first release:

![Diagram of Coordination Service](image)

**Figure 34. Structure of the Coordination Service component**

The web service exposed is based on a REST architecture, the reasons for choosing REST and not, for instance, SOAP are described in section 5.7.

The following figure depicts a Class Diagram of the Coordination Service:

![Class Diagram of Coordination Service](image)

**Figure 35. Class Diagram of the Coordination Service**
The web service is exposed through servlets, one for each resource’s type. Each servlet interacts with the database through a DBManager Object. This interface defines all methods needed to select, add, delete or update the database table. The DBManager interface is implemented by the MySqlManager class which allows the Coordination Services to access the MySql database implemented for this release of the TENCompetence client. The database is described in the DataModel section.

The XML documents exchanged with the client are managed by classes specifically designed for each kind of document (Competence, Action, Resource etc.).

The class “Config” is dedicated to the managing of the database’s connection data that are read from a configuration file.

In order to access the database data the TENCompetence client should interact with a REST service implemented through Servlets. The data sent to the servlets are XML strings. DBManager (the database manager) reads it and extracts the important data such as ID, relation etc. and then maps it to the database structure. All the other data is stored as it is received. The following is an example of this:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<user>
  <password single="true" isNull="false">pwd</password>
  <subscriptions single="false" isNull="false">
    <element isNull="false">http://localhost/server/LN/1</element>
    <element isNull="false">http://localhost/server/Action/5</element>
  </subscriptions>
  <role-levels single="false" isNull="false"></role-levels>
  <proficiency-levels single="false" isNull="false"></proficiency-levels>
  <uri single="true" isNull="false">http://localhost/server/User/user_id</uri>
  <emailconfidential single="true" isNull="false">true</emailconfidential>
  <email single="true" isNull="false">email</email>
  <name single="true" isNull="false">username</name>
  <skypeaccount single="true" isNull="false">skype</skypeaccount>
</user>
```

If the client needs to access data related to a particular user it has to send a request to the “User” servlet with the “GET” method. An example call would be:

```
http://tenchost/servlet/user/john
```

It receives the XML document above as response.

The tags with the attribute “single=true” come from the “user” table, in this specific case: email, password etc. Not all information is stored in specific fields in the user table, but only the most important, the other are stored as XML in the BLOB field. The other tags (“single = false”) come from related tables. This means that some queries are made on the database to retrieve this information.

In this example, <password> comes directly from the related field in the table “user”. It is stored in a specific field because it is frequently accessed for authentication. The child element <subscription> comes from related data stored in “learning_network” and “action”, <role-level> comes from the “role” table, whilst all other data are stored in the BLOB field of the “user” table.
On the client side if the client needs to insert a new user it should call the same servlet with the “POST” method. An example call could be:

http://tenchost/servlet/user/smith

and the data that should be passed as body of the HTTP call:

<?xml version="1.0" encoding="UTF-8"?>
/user>
    <password single="true" isNull="false">pwd</password>
    <uri single="true" isNull="false">http://localhost/server/User/user_id</uri>
    <emailconfidential single="true" isNull="false">true</emailconfidential>
    <email single="true" isNull="false">email</email>
    <name single="true" isNull="false">username</name>
    <skypeaccount single="true" isNull="false">skype</skypeaccount>
</user>

During the insertion, the information related to the other table should not be passed. To add a user to a learning network, the specific servlet should be called.

5.5.1. Servlets for the Coordination Service

The Service is based on a set of servlets each of which is able to respond to a determined set of HTTP requests as per the REST approach. The idea is to have a servlet for every component of the service, thus we have one servlet for the resources, one for the users, one for the competences, one for learning network, and so on. The several operations that a user can do, in this way, are subdivided to the various servlets that work independently of each other.

5.5.2. General considerations for the Coordination Service’s servlets

For each servlet the client should make an HTTP request using the standard HTTP methods (GET, DELETE, PUT, POST).

The structure of the call is

http://tenchost/.../servletname/[IDxxx]?param1=x&param2=y&....

Where:

- *servletname* is the name of the servlet.
- *[IDxxx]* is the ID of the resource (a “resource” in this context refers to any kind of data like action, resource, role etc.) that should be retrieved. It is optional.
- *?param1=x&param2=y&....* is the list of parameters

The call without IDxxx supports only GET, and POST operations. If GET is used the servlet returns a list of all resources, if POST is used a new resource is created.
If IDxxx is appended, the resource with IDxxx will be the target of the HTTP call.

In general the action performed depending on the HTTP method is:

1. **GET**: retrieves a resource.
2. **DELETE**: deletes a resource.
3. **POST**: creates a resource.
4. **PUT**: updates a resource.

If some parameters are passed to the HTTP call the action performed could be restricted or modified; for example it could be asked only the competences of a specific user passing the `userid` as parameters to the servlet “competence”.

If an error occurs an HTTP Error is returned. For example:

1. Error 404 if a resource doesn’t exist.
2. Error 500 for a generic internal server error
3. Error 403 “Forbidden”. If a user has no permission to access the resource.

Some calls require the user to be logged into the system to determine access permissions. For this reason the user needs to log into the system by calling the servlet “UserAccess” and passing it their credentials. A cookie is returned to maintain the session information. The use of the session is not REST compliant, but the userid is the only information stored and it’s important for the TENCompetence System to expose the service to the WWW and avoid unauthorized actions. Other solutions can be explored to avoid the use of session but security issues are not a priority for the prototype.

Each servlet implements four methods:

1. `doPost()`
2. `doGet()`
3. `doDelete()`
4. `doPut()`

The next section describes in detail which servlets are implemented for the prototype and their supported operations. For each servlet a description table is provided. If the `parameters` column contains a parameter within parenthesis this means that the parameter is included in the body of the request, otherwise it should be passed appended to the URI. The last column of each table indicates whether the URI contains the ID of the resource.

### 5.5.3. UserAccess

This servlet controls access to the user’s service. Through it, a new user can be enrolled to the service by inserting one’s own id and password, and the file xml that contains its information.

Registered users can access, sending id and password, or exit the service.
This servlet and the User servlet contain plain text passwords in the URL, which is not a good solution. This will be improved before the Antelope release. The descriptions in this document currently match with the preliminary prototype.

### 5.5.4. User

Through this servlet, a user can update or recover their own data, remove himself from the service or access the list of all users.

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
<th>Description</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>id=xx&amp;pwd=yy</td>
<td>Return the full list of Users</td>
<td>X</td>
</tr>
<tr>
<td>GET</td>
<td></td>
<td>Return the User’s xml</td>
<td>X</td>
</tr>
<tr>
<td>DELETE</td>
<td></td>
<td>Delete User from the service</td>
<td>X</td>
</tr>
<tr>
<td>POST</td>
<td>id=xx&amp;pwd=yy{xml}</td>
<td>Create new User</td>
<td></td>
</tr>
<tr>
<td>PUT</td>
<td>{xml}</td>
<td>Update the User’s xml</td>
<td>X</td>
</tr>
</tbody>
</table>

### 5.5.5. LN

This servlet allows the user to manage Learning Networks.

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
<th>Description</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>{xml}</td>
<td>Insert new Learning Network</td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td></td>
<td>Return the full list of Learning Networks</td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td></td>
<td>Return the xml description of the Learning Network</td>
<td>X</td>
</tr>
<tr>
<td>GET</td>
<td>iduser=xx</td>
<td>Return the list of user’s Learning Networks</td>
<td>X</td>
</tr>
<tr>
<td>DELETE</td>
<td></td>
<td>Delete Learning Network</td>
<td>X</td>
</tr>
<tr>
<td>PUT</td>
<td>{xml}</td>
<td>Update the Learning Network</td>
<td>X</td>
</tr>
</tbody>
</table>
5.5.6. CM
This servlet manages the competence map. The user can create or remove a competence map and retrieve the competence map of one learning network.

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
<th>Description</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>id_ln=xx {xml}</td>
<td>Insert new competence map for learning Network xx</td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td>id_ln=xx</td>
<td>Return the competence map of Learning Network xx</td>
<td></td>
</tr>
<tr>
<td>DELETE</td>
<td></td>
<td>Delete Competence Map</td>
<td>X</td>
</tr>
<tr>
<td>GET</td>
<td></td>
<td>Return the Competence Map xml description</td>
<td>X</td>
</tr>
<tr>
<td>PUT</td>
<td>{xml}</td>
<td>Update the Competence Map</td>
<td>X</td>
</tr>
</tbody>
</table>

5.5.7. Competence
This servlet allows the user to manage the competences.

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
<th>Description</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>id_cm=yy {xml}</td>
<td>Insert new Competence and add it to the Competence Map.</td>
<td></td>
</tr>
<tr>
<td>POST</td>
<td>iduser=xx</td>
<td>Add Competence to user xx</td>
<td>X</td>
</tr>
<tr>
<td>POST</td>
<td>id_cm=yy</td>
<td>Assign the Competence to the Competence Map yy.</td>
<td>X</td>
</tr>
<tr>
<td>GET</td>
<td>id_cm=yy</td>
<td>Return the full list of Competences</td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td>iduser=xx</td>
<td>Return the list of User’s Competences</td>
<td>X</td>
</tr>
<tr>
<td>GET</td>
<td></td>
<td>Return xml description of the Competence</td>
<td>X</td>
</tr>
<tr>
<td>PUT</td>
<td>{xml}</td>
<td>Update the Competence</td>
<td>X</td>
</tr>
<tr>
<td>DELETE</td>
<td>iduser=xx</td>
<td>Delete Competence from service</td>
<td>X</td>
</tr>
<tr>
<td>DELETE</td>
<td>id_cm=yy</td>
<td>Delete Competence for User xx</td>
<td>X</td>
</tr>
<tr>
<td>DELETE</td>
<td>id_cm=yy</td>
<td>Remove the Competence from Competence Map yy</td>
<td>X</td>
</tr>
</tbody>
</table>
### 5.5.8. Resource
This servlet allows the user to manage resources.

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
<th>Description</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>xml=&quot;xml description&quot; {resource}</td>
<td>Insert new Resource; parameter xml is the description of Resource that is sent as a stream attached to the request</td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td></td>
<td>Return the list of Resources that the User has permission to read</td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td></td>
<td>Return the stream of the Resource</td>
<td>X</td>
</tr>
<tr>
<td>GET</td>
<td>xml=true</td>
<td>Return the xml’s Resource description</td>
<td>X</td>
</tr>
<tr>
<td>PUT</td>
<td>{resource}</td>
<td>Update the Resource</td>
<td>X</td>
</tr>
<tr>
<td>PUT</td>
<td>{xml}</td>
<td>Update the Resource’s xml description</td>
<td>X</td>
</tr>
<tr>
<td>DELETE</td>
<td></td>
<td>Delete the Resource</td>
<td>X</td>
</tr>
</tbody>
</table>

### 5.5.9. Action
This servlet allows the user to manage actions.

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
<th>Description</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>id_ln=xx {xml}</td>
<td>Insert new Action in the Learning Network xx</td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td></td>
<td>Return xml’s Action description</td>
<td>X</td>
</tr>
<tr>
<td>GET</td>
<td>id_user</td>
<td>Return all Actions of a specified User</td>
<td></td>
</tr>
<tr>
<td>DELETE</td>
<td></td>
<td>Delete Action</td>
<td>X</td>
</tr>
<tr>
<td>PUT</td>
<td>{xml}</td>
<td>Update Action</td>
<td>X</td>
</tr>
</tbody>
</table>

### 5.5.10. Goal
This servlet allows the user to insert, update, get or delete goals.

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
<th>Description</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>{xml}</td>
<td>Insert new Goal</td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td>iduser=xx</td>
<td>Return all Goals of User</td>
<td>X</td>
</tr>
<tr>
<td>GET</td>
<td></td>
<td>Return Goal’s xml description</td>
<td>X</td>
</tr>
<tr>
<td>PUT</td>
<td>{xml}</td>
<td>Update Goal</td>
<td>X</td>
</tr>
<tr>
<td>DELETE</td>
<td></td>
<td>Delete Goal</td>
<td>X</td>
</tr>
</tbody>
</table>
5.5.11. Role
This servlet allows the user to insert, update, get or delete a role from the service.

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
<th>Description</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>id_cm=xx {xml}</td>
<td>Insert new Role to the Competence Map</td>
<td></td>
</tr>
<tr>
<td>POST</td>
<td>iduser=xx</td>
<td>Add Role to a User</td>
<td>X</td>
</tr>
<tr>
<td>GET</td>
<td>iduser=xx</td>
<td>Return all Roles</td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td>iduser=xx</td>
<td>Return User’s Role list</td>
<td>X</td>
</tr>
<tr>
<td>DELETE</td>
<td>iduser=xx</td>
<td>Delete Role</td>
<td>X</td>
</tr>
<tr>
<td>DELETE</td>
<td>iduser=xx</td>
<td>Delete Role for User xx</td>
<td>X</td>
</tr>
</tbody>
</table>

5.5.12. Permission
This servlet allows the user to manage the resource’s permissions.

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
<th>Description</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>description=xx</td>
<td>Create Permission with description xx</td>
<td>X</td>
</tr>
<tr>
<td>POST</td>
<td>iduser=xx&amp;idresource=yy</td>
<td>Give Permission to User xx on Resource yy</td>
<td></td>
</tr>
<tr>
<td>DELETE</td>
<td>iduser=xx&amp;idresource=yy</td>
<td>Remove all Permissions for User xx on Resource yy</td>
<td></td>
</tr>
<tr>
<td>DELETE</td>
<td>iduser=xx&amp;idresource=yy</td>
<td>Delete Permission on Resource yy for User xx</td>
<td>X</td>
</tr>
</tbody>
</table>

5.5.13. References and links:

5.6. What we reused
5.6.1. Objectives of the evaluation process
The TENCompetence software architecture must support the agile integration of heterogeneous tools and applications. During the first year of the project, it is necessary to deliver a “proof-of-concept” software framework that demonstrates this integration and that implements adaptations of existing components that will form the building blocks of such an infrastructure. An additional core requirement is to create an Open Source system, therefore those candidates for inclusion should themselves be Open Source components.
To fulfil these objectives, a selection process of Open Source tools suitable for integration into the TENCompetence architecture was initiated as part of the preparation activities of WP3. The aim was to obtain a list of component candidates that could be integrated in the first release of the software, or that could be used in the future providing to partners an overview of functionality/maturity/usability of existing components as well as details about their use and their integration requirements.

5.6.2. Evaluation process

The evaluation process started by defining the criteria for the selection of candidate tools to be integrated in the final infrastructure. The criteria were:

- The component must contribute to any of the R&D aspects of TENCompetence (Knowledge Resources, Units of Learning, Competence Developed Programmes or Learning Networks).
- The component must be Open Source
- The component must offer a clear integration procedure preferably based on Web Services as we are adopting a SOA architecture as the general framework
- The component must be considered “state of the art”
- The component conformed to or utilised open standards (international standards, “de facto” standards, etc.)

The next stage of the evaluation process consisted of collating an extensive list of candidates starting with those that had been developed previously by project partners, such as the Reload Learning Design Player and Editor, the Plex Personal Learning Environment from the University of Bolton, the Coppercore LD engine from the OUNL, the ASK-LDT editor from CERTH and the SLED LD player from the OU. Other candidates were considered from the following list:

- **Rich client technologies**: tools for developing rich clients such as Eclipse or facilitators such as Ajax, Flex2, OpenLaszlo, and AFLAX
- **Content management systems** such as Type3, Joomla, Exponent, Zope, Moodle and OpenCMS
- **Wiki Engines** such as MediaWiki (as used by WikiPedia) or phpWiki
- **Blog engines** such as Wordpress, b2evolution and CMS
- **Social Software** as del.icio.us, de.lirio.us and H2O
- **ePortfolio management systems** such as dotFolio, ELGG, OSP (Open Source Portfolio) and PebblePAD
- **Search Engines** such as ZEBRA or Google
- **Information- Event- Project- Management Systems** such as Stud.IP (manages events for universities and can directly link to the ILIAS3 LMS)
- **Learn Management Systems / Virtual Learning Environments** (LMS / VLE) such as ILIAS3, Shell, NIIMLE or Sakai
- **Visualization + Navigational Tools** such as Coraler (graphical navigation through visual Learning Nodes) and ROMA
- **Toolkits** such as Sweet.net (communication toolkit using IMS Enterprise Webservices with extensions) and OpenSPML
- **Tutor / Support Software** such as ATL (finds peer tutors for learners) or QAed
A limited set of tools and technologies were then selected for a more detailed analysis in order to determine the usefulness in TENCompetence of each pre-selected component/tool/technology. Then a list of tasks was created in order to provide criteria for the analysis (see the annex for an example of the form used in this analysis process):

- Determine general information: component name, website, brief functional description, owner, contact person
- Gather documentation
- Gather binaries and sources
- Determine license
- Analyse external interface(s)
- Analyse install and configuration procedures
- Determine status: How is its performance and stability? Is it being maintained or has development been abandoned?
- Usable in TENCompetence?

Most of the candidates proved to be useful but only a small number were selected for integration in the first release. However, the tools that were not selected may be of use in future releases. Moreover, some of them have provided us with interesting ideas that may be considered in the ensuing development process.

5.6.3. Selected and reused tools

Rich client technologies and Eclipse

The Eclipse development framework and Rich Client Platform ([5]) was chosen as the target platform for the TENCompetence client application. For developing and deploying rich client applications Eclipse provides the Rich Client Platform (RCP). Eclipse also includes Equinox, a component framework based on the OSGi standard, enabling the deployment of applications and components and an integrated update mechanism.

An integral component of Eclipse is the Plug-in Development Environment (PDE), providing tools to create, develop, test, debug, build and deploy Eclipse plug-ins, fragments, features, update sites and RCP products. PDE also provides comprehensive OSGi tooling, which makes it an ideal environment for component programming, not just Eclipse plug-in development. PDE is built on top of the Platform and JDT, and ships as part of the Eclipse SDK.
Plex
Many components have been taken from Plex, the prototype Personal Learning Environment developed at Bolton University ([6]) and re-used in the TENCompetence client application to date:

- Infrastructure code
- Datamodel code
- Conduits Framework – OPML, Atom, Blogger, Simpy
- Monkey Scripting tool
- Event model
- JDOM utilities
- Other utility functions

Plex contains more features that may be re-used in future releases. These include:

- RSS and Atom feed readers
- XCRI search
- 43Things search
- FOAF parsers and handlers
- Resources bookmarking

Hecate
Hecate ([7] and [8]) is a rich client Eclipse application enabling the user to actively participate in a learning network. Hecate was used as the first prototype for the TENCompetence client and server. The main features are:

- Create / Read / Update / Delete operations on learning networks, CDPs, competences, activities, simple units of learning, etc.
- User management
- Link to Skype
- Rating of items
- Client/Server model

CopperCore
CopperCore ([9]) is an IMS Learning Design Engine that supports all three levels of IMS Learning Design. It is built on a J2EE runtime engine which can be used to incorporate IMS Learning Design in any application. The target audience is system developers. CopperCore provides three APIs and a Test Suite. The main features are:

- full support for IMS Learning Design including level A, B and C
- three API's covering publication, administration and delivery of IMS Learning Design
- exposes J2EE, native Java and SOAP interfaces
- provides a validation library
- includes a command line interface to most of the API calls
- includes an example of a publication interface
- includes an example of a web delivery interface
- platform independent
- has built-in support for three relational databases (MS SQL Server/MSDE, PostgreSQL and HSQLDB)
- is ready for use with JBoss 3.2.x application server, but runs on other application servers as well
- licensed under the GNU GPL

**Reload Tools**

Reload ([10]) is a set of runtime and editing tools that facilitate the use of emerging Learning Technology Interoperability specifications such as those produced by ADL and IMS. Re-usable components that offer some level of integration are:

- IMS Learning Design Editor for Levels A, B, C
- IMS Learning Design Runtime player
- SCORM 1.2 and 2004 Editor and Player
- IMS Metadata editor

**phpBB**

The communication services are extended by integrating the client with phpBB ([11]), a high powered, fully scalable, and highly customizable Open Source bulletin board package. The TENC client offers functionality for creating / reading / updating / deleting forum posts and uses the phpBB forum software for the actual storage of all these forum posts.

To communicate using the create, read, update and delete operations of the Atom publishing protocol, phpBB and the SmartFeed ([12] and [13]) modification for phpBB were used and extended. The extension mechanism of the Atom format was used to add a few attributes to the Atom format for forum specific fields.

**5.7. Why REST in TENCompetence?**

Currently there are basically three approaches to web services:

1. **XML-RPC**: a Remote Procedure Call protocol which uses XML to encode its calls and HTTP as a transport mechanism.
2. **SOAP**: a protocol for exchanging XML-based messages over a computer network, normally using HTTP.
3. **REST**: a software architectural style for distributed hypermedia systems like the World Wide Web.

It is clear from these definitions that XML-RPC and SOAP are “protocol” based, whereas REST is a software architectural style. XML-RPC and SOAP are similar and SOAP could be considered as a successor of XML-RPC. For reasons that are out of the scope of this document, XML-RPC is considered too simple and not useful for a complex architecture leaving SOAP and REST as candidates that can fulfil the requirements of the TENCompetence System.
Each method has its advantages and disadvantages. A REST approach is advantageous in the following cases:

1. The web services are completely stateless. A good test is to consider whether the interaction can survive a restart of the server.
2. A caching infrastructure can be leveraged for performance. If the data that the web service returns is not dynamically generated and can be cached, then the caching infrastructure that web servers and other intermediaries inherently provide can be leveraged to improve performance. However, the developer must take care because such caches are limited to the HTTP GET method for most servers.
3. The service producer and service consumer have a mutual understanding of the context and content being passed along. Because there is no formal way to describe the web services interface, both parties must agree on the schemas that describe the data being exchanged and on ways to process it meaningfully. In the real world, most commercial applications that expose services as RESTful implementations also distribute so-called value-added toolkits that describe the interfaces to developers in popular programming languages.
4. Bandwidth is particularly important and needs to be limited. REST is particularly useful for limited-profile devices such as PDAs and mobile phones, for which the overhead of headers and additional layers of SOAP elements on the XML payload must be restricted.
5. Web service delivery or aggregation into existing web sites can be enabled easily with a RESTful style. Developers can use technologies such as Asynchronous JavaScript with XML (AJAX) and toolkits such as Direct Web Remoting (DWR) to consume the services in their web applications. Rather than starting from scratch, services can be exposed with XML and consumed by HTML pages without significantly refactoring the existing web site architecture. Existing developers will be more productive because they are adding to something they are already familiar with, rather than having to start from scratch with new technology.

Considering the tasks required of the TENCompetence infrastructure, REST seems to fit these circumstances very well:

1. Except for user authentication the Coordination Service doesn’t need to store any session information.
2. In the future caching could be an important issue because the TENCompetence System will be used by many people and a caching system could be very important to avoid overloading the servers and to speed up client-server interaction.
3. In the first phase the consumers of the Coordination Services are the TENCompetence clients. It is reasonable to suppose the Coordination service and the TENCompetence client are designed on same schemas or formats of the data exchanged.
4. Even if the actual client is not “mobile ready“ it’s important to consider that in a near future TENCompetence could offer their services to these devices.
5. An easy integration with other systems is important, because many different parties might use the services of the TENC system. And also within TENC System many components will use Coordination Service.
A SOAP-based design may be appropriate when:

1. A formal contract must be established to describe the interface that the web service offers. The Web Services Description Language (WSDL) describes the details such as messages, operations, bindings, and location of the web service.
2. The architecture must address complex non-functional requirements. Many web service specifications address such requirements and establish a common vocabulary for them. Examples include Transactions, Security, Addressing, Trust, Coordination, and so on. Most real-world applications go beyond simple CRUD operations and require contextual information and conversational state to be maintained. With the RESTful approach, developers must build this plumbing into the application layer themselves.
3. The architecture needs to handle asynchronous processing and invocation. In such cases, the infrastructure provided by standards such as WSRM and APIs such as JAX-WS with their client-side asynchronous invocation support can be leveraged out of the box.

When comparing these circumstances with the operations of the Coordination Service, it results in the following conclusions:

1. Should be very easy to be integrated and a formal contract and a WSDL description could complicate the integration.
2. Are very simple operations that don’t need any contextual information
3. Aren’t asynchronous operations.

When comparing the results for REST and SOAP, it is obvious REST is a better approach. What is possible with REST is also possible with a SOAP approach, but the key point here is that in the Coordination Service the greater part of the operations are “CRUD” operations and for this kind of work REST is the natural approach.
6. Deployment view

The nodes required to run the full functionality of the TENC system for Antelope can be divided into two groups:

1. TENC nodes: the main nodes for the TENC infrastructure. These nodes run the clients (Client), the coordination service (Coordination Server) and the database (DB Server).
2. External nodes: nodes running independently of the TENC infrastructure. These are existing systems used by TENC to offer additional (optional) functionality.

The Coordination Server runs Tomcat as a Servlet container to handle the HTTP requests for the coordination service. It communicates with the DB Server for the retrieval and storage of data. The Coordination Server and DB Server are shown as two nodes, but they can also run on the same node. The Coordination Server can also run concurrently on multiple nodes, in order to spread the load.

Many Client instances can run at the same time, thus allowing multiple users to use the TENC infrastructure concurrently, and these Clients can communicate with the external nodes for sharing resources to and retrieving resources from many different nodes. The Atom publishing, del.icio.us, OPML and Simpy protocols are supported for communication and all systems supporting these protocols can be used. The “AtomExt” interface is specific to phpBB (see 5.6.3 Selected and reused tools).

The following diagram shows Blogger as an example target for publishing messages using Atom. The same protocol can be used to publish to all other sites supporting the Atom publishing protocol.

![Figure 36. TENCompetence Deployment diagram](image-url)
7. Developer Guidelines

7.1. SourceForge and Bugzilla

7.1.1. Background

Development work in the TENCompetence project requires the use of a server-based solution in order to track bugs, feature requests, and enhancements. The project is already using SourceForge (http://sourceforge.net/projects/tencompetence) as a CVS repository and point of presence. The SourceForge hosting facility includes a bug tracking service. However, the TENCompetence project requires a more substantial and fully featured bug tracking service which we propose to be an externally hosted Bugzilla Service.

7.1.2. Bugzilla

Bugzilla is a “defect tracking” system that can track bugs, feature requests, and patches. In spite of its moniker, it used for more than just bug tracking. An example of a “bug” could be an item that tracks the progress of a feature request or enhancement. Bugzilla is free and customisable but has to be set up and hosted on a server. Bugzilla is used by many “big name” projects including the Mozilla Project, Eclipse, and NASA.

Features included can be found at http://www.bugzilla.org/features/

Bugzilla relies on an installed web server, such as Apache and a Database Management System, such as MySQL. Bugs and issues can be submitted by anybody, and will be assigned to a particular developer after a process of triage. Various status updates for each bug are allowed, together with user notes and bug examples.

An example of a Bugzilla bug report page can be found here - https://bugs.eclipse.org/bugs/show_bug.cgi?id=64672

Release notes such as those for Bugzilla 2.20.1 indicate the exact set of dependencies, which include:

- A compatible database server (often a version of MySQL)
- A suitable release of Perl 5
- An assortment of Perl modules
- A compatible web server such as Apache (though any web server that supports CGI can work)
- A suitable mail transfer agent such as Sendmail, Qmail, Postfix, or Exam

So, in order to set this up the project would require an available server space and a developer to install, configure and maintain the system. The documentation states that it is easier to install on a Unix-based system than on a Windows system, though it can be achieved (see http://www.bugzilla.org/docs/2.20/html/os-specific.html )
An alternative is to pay for a Bugzilla hosting service such as that found at http://wush.net/bugzilla.php. This works out at about $15 per month. This will be the preferred option.

### 7.1.3. Eclipse integration with Bugzilla provided by Mylar

Mylar is an Eclipse Technology Project that consists of a plug-in for the Eclipse platform. This plug-in provides many useful functions allowing the Eclipse user to narrow in and focus on specific development tasks:

- Provides a task focused UI for Eclipse that makes working with very large workspaces as easy as working with small ones.
- Supports task management and monitors your work activity to identify information relevant to the task-at-hand.
- Provides integrated Bugzilla support

Perhaps the best way to get a sense of what Mylar offers is to watch the Camtasia demos here:


Mylar is potentially of use to the TENCompetence development team apart from the Bugzilla integration features because of its task-oriented functionality. With Mylar, a developer can manage personal task and planning, and manage views and editors.

Bugzilla integration in Eclipse provided by Mylar means that a developer can link tasks to bugs and feature requests. The developer can quickly file a bug report from within Eclipse, query the bug base from within Eclipse and the resulting bugs and feature requests can be allocated as tasks to individual developers.

### 7.1.4. References and Links:

TENCompetence SourceForge - http://sourceforge.net/projects/tencompetence
Bugzilla - http://www.bugzilla.org
Mylar - http://www.eclipse.org/mylar
7.2. Unit Testing and JUnit

7.2.1. Background
Testing of the TENCompetence system software components is an integral part of the
development process

JUnit is a unit testing framework written for the developer who implements unit tests
in Java. It is free, Open Source and is easily integrated into the developer’s test
environment as either a standalone library or in tight integration with the Eclipse IDE.

The junit.jar library jar is linked to the code containing the tests’ classpath. JUnit is
currently at version 4.1, the previous version being 3.8.1 which is the default version
used by Eclipse 3.2. JUnit ships with many useful features:

- A variety of front-ends that can display the results of the tests – command line,
  AWT, Swing
- Separate classloaders for each unit test
- Integration with popular tools like Ant, Maven and Eclipse

7.2.2. Integration with Eclipse
The development IDE and target platform chosen for the project is Eclipse. The
Eclipse IDE workbench ships with a built in JUnit testing framework that is fully
integrated into the Java development process. This makes it extremely convenient for
developers to test their code instantly with graphical feedback in the JUnit View or by
running test suites.

Eclipse contains the following JUnit integration features:

- JUnit View provides instant feedback of passing and failing tests
- Ability to run single or multiple tests from Eclipse
- JUnit functions available from contextual menus

7.2.3. JUnit 3 or JUnit 4?
The default version of JUnit used in Eclipse is version 3.8.1. This version works with
Java versions 1.4.x and above. The current version of JUnit is 4.1 but depends on
features provided by Java version 5.0 and above in order to compile and run.

The main features of JUnit 4 are

- It Requires JDK 5 to run
- Test classes do not have to extend from junit.framework.TestCase
- Test methods do not have to be prefixed with ‘test’
- There is no difference between the old assert methods and the new assert methods
- Use @Test annotations to mark a method as a test case
- @Before and @After annotations take care of set up and tear down
• @BeforeClass and @AfterClass annotations take care of one time set up and one time tear down
• @Test annotations can take a parameter for timeout. Test fails if the test takes more time to execute
• @Test annotations can take a parameter that declares the type of exception to be thrown
• JUnit4Adapter enables running the new JUnit4 tests using the old JUnit runners
• Old JUnit tests can be run in the new JUnit4 runner

It is yet to be decided whether to use this latest version of JUnit, and it would seem logical to do so since code is being written against Java version 5.0. However, it should be noted that the current version of Ant (1.6.5), which is used for running JUnit automation scripts and reporting, does not work with JUnit 4. Therefore, if the project wishes to use Ant for automating JUnit tests and reporting it will either have to wait until Ant fully supports JUnit 4, or continue to use JUnit 3.8.1.

7.2.4. JUnit TENCompetence source folder structure and naming conventions

JUnit tests should be located in a separate module that maintains a mirror package structure of the original module that is being tested. This ensures that the main module does not have to link to or contain the junit.jar library, and also so that users can check out the code from CVS without having to get the tests as well, these remaining optional in the corresponding JUnit tests module. This is the practice that the Eclipse development team use themselves.

For example, a plug-in module called org.tencompetence.tencc would have its tests in a separate module called org.tencompetence.tencc.tests and the package structure would be as follows:

The original module:

```
org.tencompetence.tencc
 |----src
 |----org.tencompetence.tencc
 |----package1
 |----StringUtils.java
 |----package2
 |----package3
```
And the JUnit test module structure:

```
org.tencompetence.tencc.tests
 |----src
 |----org.tencompetence.tencc
 |------package1
 |--------StringUtilsTest.java
 |------package2
 |------package3
```

File names of tests should follow the naming convention of “Class to be tested” followed by the suffix “Test”. For example:

Class to be tested: StringUtils
Name of test class: StringUtilsTest

### 7.2.5. Automated and Manual Testing frameworks

A developer can run single or multiple tests from within Eclipse “on the fly” as needed. Additionally, we will be running nightly builds and automated scripts that will run all JUnit tests and report the results as HTML files. We have already created an Ant script to perform the task of running automated JUnit tests. However, as noted above, the current version of Ant is only able to invoke JUnit tasks for JUnit version 3.8.1, not version 4.

### 7.2.6. Guidance for writing tests

Providing best practice guidance for writing JUnit tests is beyond the scope of this document. However, developers are recommended to study the material available here:

JUnit Home Page - [http://www.junit.org](http://www.junit.org)
JUnit Tutorial - [http://clarkware.com/articles/JUnitPrimer.html](http://clarkware.com/articles/JUnitPrimer.html)
The New Features of JUnit 4 - [http://www.devx.com/Java/Article/31983](http://www.devx.com/Java/Article/31983)
7.3. The Use of a CVS Repository in the TENCompetence Project

7.3.1. Background
The TENCompetence development process works in an open manner, utilising an Open Source licensing model and attempting to maintain transparency at all levels. One means to accomplish this transparency is to ensure that all code, documentation and development artefacts are available in a Concurrent Versions System (CVS) Repository.

7.3.2. CVS and SourceForge
The TENCompetence project uses SourceForge for its CVS repository. This is a free service that provides hosting for Open Source projects and provides additional services such as issue tracking, forums, project presence, CVS and Subversion Repositories.

7.3.3. Access rights and user permissions
We have allocated key project staff as CVS administrators and assigned key developers with CVS developer status. By default, CVS access is read only, and this includes public anonymous access. However, developers are granted write access on a per-module basis.

In order for a developer to be granted write access to files in a module or modules, the following steps have to be taken:

1. The aspiring developer has to register at SourceForge with a user name.
2. A SourceForge administrator has to add the user as a developer to the SourceForge project admin page.
3. A SourceForge administrator has to edit the avail file to include the user name and module name(s) required for read/write access.

As of the time of writing of this document, the avail file is as follows:

```
unavail
avail|hubertvogten,harriemartens,rulem,phillipus,scottwilson
avail|arnewolf2|org.tencompetence.tencc.questionauthoring
avail|dicerto|org.tencompetence.tencs
avail|dicerto|org.tencompetence.tencs.coordination
avail|dicerto|org.tencompetence.tencs.policy
avail|ps3com|wp6/org.tencompetence.imsmodel
avail|ps3com|wp6/org.tencompetence.imsmodel.tests
avail|ps3com|wp6/org.tencompetence.ldauthor
avail|ps3com,pzervas,mloizos,pythk|wp6/org.tencompetence.ldauthor.transform
avail|ps3com|wp6/org.tencompetence.xstream
```
The first line, “unavail”, ensures that by default all developers do not have write access to the repository. The line “avail” is followed by user name(s) and the name of the module that is given write access. Full documentation for editing the avail file is available at the SourceForge website.

7.3.4. Integration with Eclipse
The development IDE and target platform chosen for the project is Eclipse. The Eclipse IDE workbench ships with a built in CVS client which is fully integrated into the Java development process. This makes it extremely convenient for developers to synchronize their workspace with the CVS repository and to monitor history, annotations and file comparisons within the IDE. Thus, no other CVS client is needed to perform all common CVS tasks.

7.3.5. Repository details
The SourceForge CVS connection details are as follows:

Web Page: http://sourceforge.net/cvs/?group_id=159487
Host location: tencompetence.cvs.sourceforge.net
Repository Path: /cvsroot/tencompetence
Connection type: extssh
Public User Name: anonymous

7.3.6. Modules and folder structure
All code is deposited under the “HEAD” CVS location. Work Package 3 code is placed at the top level under HEAD, while other Work Package code is placed in appropriately named subfolders (“wp6”, “wp7”, “wp8”, and so on).

The naming convention for a code module for the TENCompetence client application is the project namespace (org.tencompetence.tencc) followed by the module name, as follows:

org.tencompetence.tencc.modulename

Where “modulename” is an appropriate name of the module.

As an example, if one had a module named "org.tencompetence.myproject" that came under the remit of WP5 then a developer using Eclipse would perform the following steps:

1. Right-click on the Project in the Package Explorer in Eclipse
2. Choose Team->Share Project...
3. Select the 10Competence Repository
4. Choose "Use specified module name" which would be wp5/org.tencompetence.myproject
7.3.7. JUnit folder structure

JUnit tests relating to a module should be organised in a separate module that maintains a mirror package structure of the original module. This ensures that the main module does not have to link to or contain the junit.jar library, and also so that users can check out the code without having to get the tests as well, these remaining optional in the corresponding JUnit tests module. This is the practice that the Eclipse development team use themselves.

For example, a plug-in module called org.tencompetence.tencc would have its tests in a separate module called org.tencompetence.tencc.tests and the package structure would be as follows:

The original module:

```
org.tencompetence.tencc
|----src
|----org.tencompetence.tencc
|----package1
|----package2
|----package3
```

And the JUnit test module structure:

```
org.tencompetence.tencc.tests
|----src
|----org.tencompetence.tencc
|----package1
|----package2
|----package3
```

For more information on JUnit and package structures, see the section on Unit testing.

7.3.8. Other considerations when using folders

The "bin" folder and any folders that contain compiled code files (*.class files for Java) should be excluded from the CVS, as it should only contain source files and binary library files.

3rd party libraries can be uploaded to CVS (together with their licences) provided that a version number is used as part of their file name or set in the Eclipse manifest if it is delivered as a “wrapped” Eclipse plug-in. Versions of popular libraries can vary enormously. JDOM 1.0 is *very* different to JDOM 0.9. The same goes for Tomcat - some things work fine for 5.5, but not for version 5.0. Name the file jdom-1.0.jar, not jdom.jar.

Note that modules can not be deleted permanently by users. For this to happen a formal request has to be submitted to the SourceForge technical support team by a project administrator. This process can take a few days to complete.
7.3.9. House Rules

In order to maintain good housekeeping for using the CVS repository in the project the following rules have been put in place:

1. Committed code should at all times be able to be compiled.
2. Committed code should at all times be able to be run.
3. Developers should check out the latest code from CVS before committing their own changes.
4. Any code conflicts should be resolved between developers preferably informally, but failing that via a project co-ordinator.
5. Folders (a.k.a "modules") should not be created under CVSROOT but only directly under HEAD. See above for more details as to folder structure.
6. Module naming conventions should be followed, see above.
7. JUnit tests should be given their own modules, see above.
8. Compiled class files and the compiled output folder (usually “bin”) should not be added to the CVS. Only source files and libraries can be uploaded.
9. Library files should be named according to their version and/or the Eclipse manifest version number set accordingly.

7.3.10. Component Owners

Each component or other coherent piece of software in the TENCompetence system will be assigned a Component Owner. The Component Owner is the organisation, university, or company that created it. When a component requires changes from a non-Component Owner, any changes should be discussed with the Component Owner first. The Component Owner decides how to handle changes. For example, a developer working on a QTI tool that requires changes to the central data API, would have to contact Harrie Martens or Hubert Vogten. Depending on the required changes Harrie or Hubert could either:

- update the central data API, or
- allow the developer to make the changes herself, or
- allow the developer to submit a patch file

7.3.11. References and Links:

CVS Home Page - http://www.nongnu.org/cvs/
SourceForge and CVS - http://sourceforge.net/docs/E04/
TENCompetence SourceForge - http://sourceforge.net/projects/tencompetence
Eclipse – http://www.eclipse.org
7.4. Coding conventions and guidelines

Note - the following are modelled on the Eclipse / Sun coding conventions and guidelines.

7.4.1. Naming conventions

Classes and Interfaces
Class names should be nouns, in mixed case with the first letter of each internal word capitalized. Try to keep the class names simple and descriptive. Use whole words - avoid acronyms and abbreviations (unless the abbreviation is much more widely used than the long form, such as URL or HTML).

Examples:
- class Raster;
- class ImageSprite;

Interface names should be capitalized like class names. For interface names, we follow the "I"-for-interface convention: all interface names are prefixed with an "I". For example, "IWorkspace" or "IIndex". This convention aids code readability by making interface names more readily recognizable.

Additional rules:

The names of exception classes (subclasses of Exception) should follow the common practice of ending in "Exception".

Methods
Methods should be verbs, in mixed case with the first letter lowercase, with the first letter of each internal word capitalized.

Examples:
- run();
- runFast();
- getBackground();

Additional rules:

The naming of methods should follow common practice for naming getters (getX()), setters (setX()), and predicates (isX(), hasX()).
Variables

Instance, static, and class constants are in mixed case with a lowercase first letter. Internal words start with capital letters. Variable names should not start with dollar sign $ characters. Variable names should be short yet meaningful. The choice of a variable name should be mnemonic - that is, designed to indicate to the casual observer the intent of its use. One-character variable names should be avoided except for temporary "throwaway" variables. Common names for temporary variables are i, j, k, m, and n for integers; c, d, and e for characters.

Examples:

- int i;
- char c;
- float myWidth;

Constants

The names of class constants and of ANSI constants should be all uppercase with words separated by underscores ("_").

Examples:

- static final int MIN_WIDTH = 4;
- static final int MAX_WIDTH = 999;
- static final int GET_THE_CPU = 1;

Plug-ins and Extension Points

All plug-ins (and plug-in fragments) must have unique identifiers following the same style of naming convention as Java packages. For example, the Eclipse workbench plug-in is named org.eclipse.ui.

Extension points that expect multiple extensions should have plural names. For example, "builders" rather than "builder".

Further guidelines as specified by Sun can be found here -
7.4.2. Conventions for plug-ins

Shared code should be packaged into plug-ins so that each plug-in offers a unique set of functionality allowing for optimal re-use. A proposed core plug-in structure is as follows:

1. TENCompetence Client Host
2. General Utilities Plug-in
3. GUI Utilities Plug-in
4. Persistence / File handling Plug-in
5. XML / JDOM Utilities Plug-in
6. Conduits/Co-ordination Framework Plug-in

Dependencies between UI components and business logic components should be eliminated. For example, utility type plug-ins should be factored such that they do not depend on or reference UI components. A Model-View-Controller architecture and Event Model facilitates this pattern. References to Java Interfaces rather than classes also contributes to the plug-in architecture.

Care should be taken when exporting packages and classes. Public packages and classes can be declared in the plug-in manifest and these should usually be limited to public API packages. All other implementations can be made private. This requires the separation of the API from internal implementation. By exporting only API classes one can limit the visibility to downstream clients. Negotiation between plug-in developers may be necessary in order to determine the right balance between public and private dependencies, and developers should reveal only the API in which they have confidence, but be prepared to reveal more of the API as clients ask for it.

Re-usable third-party libraries should be wrapped in plug-in manifests for deployment as Eclipse plug-ins and named accordingly. For example, the XML library JDOM 1.0 could be packaged with a manifest file and made available as org.jdom.plugin version 1.0.0. Plug-ins that depend on a particular third-party library may instead ship the required library file(s) in the packaged plug-in.

JUnit tests for each component should be provided in their own plug-in modules – see section 7.2 for details.

Extension Points

Developers are encouraged to provide and declare Extension Points in their plug-in manifests. This Eclipse mechanism provides an XML-based schema approach for declaring the mechanics of a plug-in’s extension point(s).

Extension points define new function points for the platform or application that other plug-ins can plug into. An Extension Point is declared in XML-based schema file that defines the grammar that formally expresses elements, attributes, and types that it relates to. This information can be used by plug-ins to validate extensions. An example of an Eclipse Extension Point is a Menu Item, a View, or an Editor.
Existing Extension Points in the TENCompetence Client are:

org.tencompetence.tencc.datamodelfactory
org.tencompetence.tencc.datamodelproviders
org.tencompetence.tencc.conduits.sharing_conduits
org.tencompetence.tencc.coordination.providers

These are described in more detail elsewhere in this document.

Future plug-in contributors may then leverage these extension points to provide further functionality to the client.

**Naming conventions**

TENCompetence plug-ins must follow the naming convention of project namespace followed by module name. The project namespace is org.tencompetence. TENCompetence client plug-ins have the “tencc” name appended to the namespace. For example, a new plug-in for the client application would be named org.tencompetence.tencc.myplugin.

The convention for naming private and public API packages is for every plug-in intended for external use to have two name spaces. If a package name contains internal, the classes it contains are not intended to be used publicly.

### 7.4.3. User Interface Guidelines

Again, these are modelled after the Eclipse User Interface Guidelines and being too lengthy to reprint here can be found at:


### 7.5. Runtime, Platforms, Tools and Libraries

**Java 5** (v1.5.0) is to be used for the SDK and target runtime. Java 6 (v1.6.0) may be used at a later date once it has been firmly proved and established on all platforms (Windows, Linux, Mac).

The following table lists the **Eclipse tools and libraries** that are to be used in the project. Note that versions may be changed as updates are made available:

<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
<th>Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eclipse Platform</td>
<td>3.2.1</td>
<td>Eclipse SDK and RCP</td>
<td><a href="http://www.eclipse.org">www.eclipse.org</a></td>
</tr>
<tr>
<td>Eclipse Web Tools</td>
<td>1.5.1</td>
<td>Eclipse Web Tools plug-ins</td>
<td><a href="http://www.eclipse.org/wtp/">www.eclipse.org/wtp/</a></td>
</tr>
<tr>
<td>GEF</td>
<td>3.2.1</td>
<td>Graphical Editing Framework</td>
<td><a href="http://www.eclipse.org/gef/">www.eclipse.org/gef/</a></td>
</tr>
<tr>
<td>EMF</td>
<td>2.2.1</td>
<td>Eclipse Modelling Framework</td>
<td><a href="http://www.eclipse.org/emf/">www.eclipse.org/emf/</a></td>
</tr>
</tbody>
</table>

**Table 6. Eclipse tool and libraries used in the project**
The following table lists the **third-party tools and libraries** that are currently used in the project. Note that versions may be changed as updates are made available:

<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
<th>Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDOM</td>
<td>1.0</td>
<td>Java XML Modelling library</td>
<td><a href="http://www.jdom.org">www.jdom.org</a></td>
</tr>
<tr>
<td>Commons Logging</td>
<td>1.5.1</td>
<td>Logging services</td>
<td><a href="http://jakarta.apache.org/commons/logging/">http://jakarta.apache.org/commons/logging/</a></td>
</tr>
<tr>
<td>Commons Codec</td>
<td>3.2.1</td>
<td>Implementations of common encoders and decoders such as Base64, Hex, Phonetic and URLs</td>
<td><a href="http://jakarta.apache.org/commons/codec/">http://jakarta.apache.org/commons/codec/</a></td>
</tr>
<tr>
<td>Commons HTTP</td>
<td>2.2.1</td>
<td>Client side of the most recent HTTP standards and recommendations</td>
<td><a href="http://jakarta.apache.org/commons/httpclient/">http://jakarta.apache.org/commons/httpclient/</a></td>
</tr>
<tr>
<td>Google API</td>
<td>latest</td>
<td>Google search client</td>
<td><a href="http://code.google.com/">http://code.google.com/</a></td>
</tr>
<tr>
<td>Skype API</td>
<td>latest</td>
<td>Skype client</td>
<td><a href="https://developer.skype.com/">https://developer.skype.com/</a></td>
</tr>
<tr>
<td>Blowfish library</td>
<td>2.1.4</td>
<td>Encryption services</td>
<td><a href="http://www.schneier.com/blowfish.html">http://www.schneier.com/blowfish.html</a></td>
</tr>
<tr>
<td>Rome</td>
<td>0.8</td>
<td>RSS and Atom Feed parsers</td>
<td><a href="https://rome.dev.java.net/">https://rome.dev.java.net/</a></td>
</tr>
</tbody>
</table>

Table 7. Third-party tools and libraries used in the project

Additional libraries and tools will be specified and recommended as the project proceeds.
8. References


9. Appendices

The four attached appendices contain:

1. The TENCompetence Domain Model: contains the first release of the TENCompetence Domain model. It is expressed as an UML class diagram and a vocabulary defining each concept (class) in the model. The TENCompetence Domain Model has been created by WP2.

2. Initial Requirements: this contains high and mid level requirements for the TENCompetence Infrastructure, divided into two main parts. The first part covers general functional requirements from a users perspective and the second part provides non-functional requirements for the infrastructure. Furthermore, it covers the distribution of these requirements across the different research and development activities of TENCompetence. The Initial Requirements have been created by WP2 and were also delivered as internal deliverable ID2.1.

3. Functional description of the Scripting Monkey plug-in. The description itself doesn’t influence the architecture, but is interesting as information about the system’s plug-ins.

4. Functional description of the QA plug-in. The description itself doesn’t influence the architecture, but is interesting as information about the system’s plug-ins.
9.1. Appendix 1 - The TENCompetence Domain Model (Version 1.0)

This document contains the first release of the TENCompetence Domain model. It is expressed as an UML class diagram and a vocabulary defining each concept (class) in the model. The Domain Model serves several functions in the project:

a) to define the scope of the project, including the scope for the use cases,
b) to define the vocabulary used,
c) to define the relationship between the concepts used,
d) to define the overall conceptual architecture,
e) to provide a technological theory for the project that must be tested in the pilots,
f) to provide a starting point for the design of other models, like the data model and services,
g) to define the minimal functional components that must be present in the TENCompetence infrastructure.

The TENCompetence System that we are developing must meet at least 7 core functional requirements according to the project plan:

1. Support for new, promising, innovative pedagogical and organisational approaches for lifelong competence development that use the possibilities of new technologies available. This includes an integration of formal and informal learning.

2. Help learners to get an overview of all the possible formal and informal knowledge resources, units of learning, programmes and learning networks that are available, and to identify the most appropriate for their needs and background.

3. Stimulate the pro-active sharing of knowledge resources.

4. Provide support for competence assessment, including the assessment of the competences of applicants, employees and learners who have studied and worked in a variety of formal and informal settings.

5. Provide effective and efficient support for users during the performance of the various tasks in various roles (learner, teacher, assessor, etc.).

6. Provides support for decentralized, self-organized and empowered management.

7. Integrates four different types of models and tools used for competence development, i.e. tools and models for:
   a) knowledge sharing & management,
   b) the creation & use of learning activities and units of learning,
   c) creation & use of formal and informal competence development programmes for lifelong learning and d) creation and use of learning networks & learning communities for lifelong learning.

The model is a further elaboration of the initial domain model that was specified in the project plan. It includes more details than the original version and it concentrates on the aspects of the system that will be represented in the TENCompetence System to fulfil the 7 requirements.
All primary use cases are connected to the 'goal' class in the Domain Model. In the current description a lifelong learner can use the system to get support for the attainment of the following goals:

1. I want to keep up to date within my existing function or job
2. I want to study for a new function or job or improve my current job level
3. I want to reflect on my current competences to look which functions and jobs are within my reach or to help me define new learning goals
4. I want to improve my proficiency level of a specific competence
5. Want some support on a non-trivial learning problem
6. Want to explore the possibilities in a new field (learning network) to help define new learning goals

The model was drawn with the UML tool 'MagicDraw' version 11.0. The documentation in this document is a copy of the documentation in the MagicDraw document.

The most recent version of the Domain Model can be found at http://hdl.handle.net/1820/649
More information about TENCompetence at: www.tencompetence.org
Please provide comments to the author: rob.koper@ou.nl

1. The class diagrams

The domain model consists of three diagrams:

a. The core domain model (figure 1). These concepts are also explained in the vocabulary.
b. The rating mechanism that is available on all classes (figure 2).
c. Some packages that should be explored for future integration conform the project plan (figure 3).
Figure 1. The Core Domain Model
Figure 2. Rating

Figure 3. Packages with functionality to be elaborated in future releases
2. The vocabulary

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>When a certain event occurs (e.g., the actor wants to solve a problem), the actor searches for an adequate action to perform. The selected action plan can be any of three types: 1. A single activity (synonym: task) 2. A unit of learning 3. A competence development programme The (optional) objectives of an action are defined in terms of proficiency levels of competences or in terms of the level of a specific function/job. When the action is completed one can infer automatically that the competence is mastered. Sometimes a certificate can proof this accomplishment (part of the competence assessment). Prerequisites can be defined in two ways: a. By pointing to one or more other actions that are expected to be completed before this action can be performed. b. By pointing to competences that have to be mastered before the action can be done. This depends on the situation. In formal, designed learning networks the competence links would be the best solution. In informal learning the competence definitions could be absent, so a link to an action is needed. Actions can have time constraints, e.g., a course runs from September until December. This can be modeled with the optional attributes start-time and end-time. Actions can be blogs (or automatically logged in case of CDP tracks) of the activities, units of learning or CDPs that an actor has performed. In that case the action should be tagged as ‘blogged’ with the appropriate attribute.</td>
</tr>
<tr>
<td>activity</td>
<td>Activities are descriptions of an action that an actor has to perform (or has performed) to meet some objectives, given some prerequisites. Activities (synonym: tasks) contain an activity description to tell an actor what actions to perform. It also points to the resources and services that are needed to perform the activity. There are different types of activities: a. Learning Activities b. Support Activities (tutoring, support students, etc.)</td>
</tr>
</tbody>
</table>
### Class Name Documentation

c. Assessment activities

Activities can be completed in different ways:

a. For assessment activities: when the test is done. (e.g. all test items are answered and a score is calculated).

b. For single learning & support activities: when the actor indicates that he has completed the activity/

c. For activities embedded in a Unit of Learning more complex, property/test based completion criteria can be set.

d. Learning activities that are blogs of the activities that an actor has performed are always completed (per definition).

| actor | An actor is a person, a team of persons or a formal or informal organization. In our first implementations we focus on individuals as actors. An actor can have various roles in the learning network that can change according to the policies, but in principle each actor can have the right to perform any task in the system. This means that a person can have the role of learner and of teacher or assessor.

The major roles for an actor are:

- learner

- support provider (eg teacher, tutor, mentor, peer tutor, etc.)

- assessor

- author

Learners have goals (eg. want to improve a competence). After selection of a specific action, they perform the action. When there is no adequate action available they can make the goal `pending`. Other actors (in the role of author) can provide an adequate action.

Support providers will answer questions that learners have posted for other actors to answer.

An assessor can be involved in the competence assessment, an assessment activity or the unit of assessment.

Authors edit the learning network, the actions within the network, the competence map, the topics and the knowledge resources. |

| assessment activity | This type of activity represents classical tests (formative and summative).

Examples are:

- Multiple choice tests to test whether you master certain principles |
<table>
<thead>
<tr>
<th>Class Name</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>drill and practice tests</td>
<td>- drill and practice tests to train certain skills</td>
</tr>
<tr>
<td>Intake questionnaires to assess personal preferences for personalization purposes.</td>
<td>- Intake questionnaires to assess personal preferences for personalization purposes. In this case the assessment activity can contain one or more test items, including the mechanism to score the test (only straightforward scores like percentage good answers; more complicated schema's can be applied in the unit of assessment). The assessment activities can export to and from IMS QTI. Note: other kind of stimuli or tasks that are used in the context of a unit of assessment, e.g. the task to deal with incoming mail in an assessment center can be modeled as learning activities and included in a unit of assessment.</td>
</tr>
<tr>
<td>assessment result</td>
<td>This class represents the assessment results. There are many types of assessments, the classical assessments activities end up in a score.</td>
</tr>
<tr>
<td>assessment spec</td>
<td>A new testing/assessment spec will be developed in TENCompetence. The position at the moment about this is the following: a. The QTI will be used to model classical test items (MC, short answer, etc.). In the domain model these are mainly parts of the test activities (although also non-QTI test activities exist!). b. The more advanced, newer forms of assessment like peer assessment, performance assessment, 360 degree feedback, adaptive testing, etc. are in principle workflows in which classical test items do not play a major role (probably even non at all). The hypothesis is that these types of tests can be modelled using IMS Learning Design. The assessment spec so will probably be a reference model and a standards profile on a set of standard (mainly QTI and LD).</td>
</tr>
<tr>
<td>common competence interoperability framework</td>
<td>This is a Common Framework for Competences that has to be defined by CEN/ISSS, IEEE and others. TENCompetence can provide valuable input to this process, but the standardisation activities itself are not part of the scope of the project (is also impossible because the project is not a standardisation organisation). The Common Framework should build on the current specs from HR-XML format and IEEE RCDEO. Note: the syntax of the competence format is similar to the</td>
</tr>
<tr>
<td><strong>Class Name</strong></td>
<td><strong>Documentation</strong></td>
</tr>
<tr>
<td>----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>competency formats used in the open standards. The field EVIDENCE can be filled with the evidence that is stored in the competence assessment (typical these are ‘results’).</td>
<td></td>
</tr>
<tr>
<td>communication &amp; collaboration facilities</td>
<td>Actions contain/use communication &amp; collaborative facilities (‘services’) like: * chats * wikis * forums * VOIP * Shared white boards * Etc. In principle it should be possible to connect to any open service offered at the internet. We will provide some basic support for the most basic collaborative services needed (chats, wikis, forums, VOIP through skype).</td>
</tr>
<tr>
<td>competence</td>
<td>A competence is defined as the ability (‘disposition’) of an actor to act effectively and efficiently upon the events in an ecological niche (an occupation, a hobby, a market, a sport, etc.). In short: the ability to perform effectively in a situation. Competence is an underlying characteristic (a latent variable or a trait) that cannot be measured directly: competence is estimated from the results of the actions (performance). The basic challenge in competence modelling is to keep it as simple as possible: it is possible to have very advanced and detailed views on competences. We will try to start with the simplest model as possible (to stimulate ease of use for end-users) and depending on validation excursuses we will refine the model when there is an absolute and direct need. A competence is structured very simple as: a. a name b. a worldwide unique ID c. a competence description (free text) d. a creator (the actor who has specified the competence) e. a competence-type. This is a free text attribute to define the type of competence. Examples are: - cognitive competence, transversal competence, functional competence, etc. Another example: Knowledge, Skill, Ability, Tasks, etc. These attributes can be used to present the competences to the user in a structured way. A competence must have at least one or more proficiency</td>
</tr>
</tbody>
</table>
### Class Name Documentation

<table>
<thead>
<tr>
<th>Competence assessment</th>
<th>Competence assessment is the process in which the proficiency levels of the competences of an actor are estimated, given the historical results of his/her actions.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In this process the results of the actions of an actor (the ‘performances’) are interpreted to estimate the proficiency level of each competence and to apply the function/job level accomplishment rules to determine whether a person meets the criteria for a function or job.</td>
</tr>
<tr>
<td></td>
<td>There are different phases in assessment:</td>
</tr>
<tr>
<td></td>
<td>a. Identifying the competences (given certain function/job) that have to be estimated.</td>
</tr>
<tr>
<td></td>
<td>b. Gathering evidence (eg by using tests, by asking diploma’s, etc.) for the competences</td>
</tr>
<tr>
<td></td>
<td>c. Making the decision on the proficiency levels an actor has acquired</td>
</tr>
<tr>
<td></td>
<td>d. Making a decision whether a person complies to the requirements of the different function/job levels to determine at which role level he/she functions.</td>
</tr>
<tr>
<td></td>
<td>The role of assessor can be fulfilled by any actor (also the person himself, e.g. self-assessment) dependent on how formal the assessment must be. Methods of assessment can vary from adaptive testing, 360 degree feedback, performance assessment, peer assessment, portfolio assessment. A special case is the situation in which an action has a competence level as its objective. In that case, completing the action automatically will set the proficiency level.</td>
</tr>
<tr>
<td></td>
<td>Example of estimation of proficiency levels:</td>
</tr>
<tr>
<td></td>
<td>For instance when a person has successfully written a scientific article in an English journal (as the result of an action), and the competence map contains the competences for the domain Psychology, role Researcher (role/job level: postdoc):</td>
</tr>
<tr>
<td></td>
<td>- Be able to write an academic publication (target proficiency level = 2)</td>
</tr>
</tbody>
</table>
Then the function/job accomplishment rules can be applied to determine whether the person meets the criteria for any of the defined function/job levels.

The competence assessment can have different forms:

a. When an actor has registered a set of results in his portfolio (acquired in different learning networks), the competence assessment has to map these activities to estimate the proficiency levels of the competences in one of the competence maps in the current learning network. The results provide the evidence/proof for the latent competence. This is a process that contains human activity. The process to estimate the proficiency levels at a given competence map is called ‘positioning’.

b. When an actor has performed a designed action (‘closed world’ situation) that has as its objective a proficiency level that is defined in the competence map of the learning network, then the assessment confirms that the competence is mastered at a certain level. When the actor previously had a level that was lower than the level of the objective, the level can be increased automatically.

c. When there is a learning network with many actions, but with no competence map defined, there is no competence assessment function.

In this situation we can try to derive a competence map by clustering the types of actions into groups (e.g. by using LSA type of techniques).

**Rule for hierarchical competences**

When a competence contains a competence to represent a hierarchy, the rule is as follows:

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Be able to write in English (target level = 2)</td>
<td></td>
</tr>
<tr>
<td>- Be able to present a publication at a scientific conference (target level = 2)</td>
<td></td>
</tr>
<tr>
<td>Then the assessment determines the estimated proficiency levels of all these competences, based on the results of actions (the positioning services has created the first estimates of these values when available, default these levels have the value zero). So based on a competence assessment procedure the estimated values for the person may be the following:</td>
<td></td>
</tr>
<tr>
<td>- Be able to write an academic publication (level=2)</td>
<td></td>
</tr>
<tr>
<td>- Be able to write in English (level=1)</td>
<td></td>
</tr>
<tr>
<td>- Be able to present a publication at a scientific conference (level=0)</td>
<td></td>
</tr>
</tbody>
</table>

Then the function/job accomplishment rules can be applied to determine whether the person meets the criteria for any of the defined function/job levels.
The levels of all the sub-competences must be attained at least at the specified target level in order to conclude that the parent has been attained at the specified level. Example:

Competence A with target level = 3; subcompetences:
A1 (target level = 1)
A2 (target level = 2)

When a person has A1 at least at level 1 and A2 at least at level 2, then it can be concluded that competence A has level 3. Note: this doesn’t exclude the possibility that there is a direct assessment of competence A without measuring A1 and A2 first. In that case the levels for A1 and A2 are estimated as having their target levels.

Rule for function/job accomplishment

The requirements for a function/job level are met by an actor when the following (simple rule) is met:

An actor meets the requirements for the function/job level when he has attained a valid proficiency level for all the competences that are specified in the function/job level that is equal or higher than the specified target-proficiency levels.

E.g.:
Job level X requires:
- Competence A at level 3
- Competence B at level 2
- Competence C at level 0 (not required for this role)

Person has the following valid competences:
- Competence A at level 4
- Competence B at level 2
- Competence C at level 4

In this situation, applying the rule above, the requirements for the Job level are met.

Notice that the rules can be more difficult in future implementations. E.g. when it is allowed for a function/job to have some compensating activities (e.g. in the above example: when the person has A=2 he doesn’t meet the requirement, but
## Class Name Documentation

**C=4** can compensate for this. For the time being we apply the simple model.

### competence assessment result

A special kind of assessment result is the result of a competence assessment. This class represents the proficiency levels that a specific actor has attained after completion of a competence assessment. It contains at least:

a. The proficiency level on a competence that an actor has attained.
b. The date that the proficiency level was attained.
c. The assessor who has made the final decision about the proficiency level of the competence (e.g. the actor self or a testing agency, a university, etc.).
d. The validity of the assessment procedure used (optional); this is expressed as a percentage.
e. The reliability of the assessment procedure used (optional); this is expressed as a percentage.
f. Reference to the assessment procedure (optional). This is a url to a description of the assessment procedure.

This can also be used to set the change rights on the result.

### competence development programme (CDP)

A Competence Development Programme (CDP; synonyms: route, learning path, curriculum, programme) is an ordered set of activities and units of learning that have to be (or are) followed to attain a certain proficiency level or the requirements for a certain function/job. E.g. the courses and sequence of the courses to be followed in order to get a masters programme in psychology.

Routes can be designed (e.g. a masters programme at a university) or shared tracks (a person with a successful track shares this with other persons. These shared tracks are mostly single linear pathways).

Routes can have zero or more proficiency levels of competences as its objective OR can have zero or more function/job levels as its objective. Designed routes have probably a function/job level as its objective (e.g. a route to become a master in psychology).

Routes can be defined on a time schedule.

### competence map

A competence map contains a structured way to represent competences, competence levels and the functions/jobs. A competence map has an owner (e.g. the OUNL, University of Amsterdam, FBM-UPF, Bolton or a consortium, a standards organisation).

Theoretically a learning network can have zero or more competence maps and competence maps can be nested to model...
<table>
<thead>
<tr>
<th>Class Name</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>a competence map for each subdomains in the domain. The same actions in the network can be connected to competences in multiple maps. E.g. in a network there is a map from the OUNL (=specified using the creator attribute), but at the same time, there is a map of Sofia University. The mapping can be different, but the underlying actions can be shared.</td>
<td></td>
</tr>
</tbody>
</table>

To simplify the first implementations (and also the user experience) we have restricted this functionality in the sense that only one competence map may be present in a learning network. When multiple maps are needed we can model this for the time being using the function/role in domain concept. E.g. in the digital cinema pilot UPF can have the role ‘UPF operator’ and Unv. of Sofia the role of ‘UniSofia operator’ each with different competences defined. |

**Example Competence Map**

LN: Domain:=Medical Profession; Owner:=University of Amsterdam Competence Map:

Domain:=Medical Profession; Owner:=University of Amsterdam

Function/job: Doctor at Expert level
Competence1: diagnosis (level 3)
Competence1.1: using a stethoscope (level 3)
Competence1.2: taking an anamnese (level 3)
Competence2: intervention/treatment (level 3)
Competence3: bed care (level 1)

Function/job: Nurse at Expert level
Competence 3: bed care (level 3)
Competence 3.1: etc.

In this example some of the competences are shared for different functions/jobs.

Because of technical reasons (mainly authorisation) a competence map is part of exactly one learning network. This means that a change in a map that is shared among different LNs should be made to every instance of the map in every LN. Changes to a shared map cannot be propagated to all the other using Networks (because than the owner should be authorised in all the other networks). The import/export mechanism is of help to support this process.
<table>
<thead>
<tr>
<th>Class Name</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>competence observatory</td>
<td>The competence observatory is a function in which the actors of a profession or knowledge domain discuss and decide upon the competences/proficiency levels per function/job level that are relevant in the domain. This includes the definition of the assessment criteria for the different proficiency levels. The outcome of the competence observatory is imported as a competence map, using an interoperable competence description format from the Common Competence Interoperability Framework). This map can then be imported in the competence map of a learning network.</td>
</tr>
</tbody>
</table>
| eportfolio spec            | The portfolio is not a separate entity in the system, but a view on the data in the system for different purposes, e.g.:  
  a. to reflect on ones accomplishments  
  b. to make a profile that can be published to the LN  
  c. to make a profile (a CV) that can be published to a specific group (e.g. an employer).  
These views can be printed and exported (to be used in another ePortolio system) in a standard format like IMS ePortfolio (however this seems not a very adequate spec).  
The data that are typically of interest in an ePortfolio view are:  
- stopped not completed actions  
- completed actions  
- results of completed actions  
- results of the competence assessments of completed actions (evidence for having attained a competence in the learning network)  
- data stamps for all data (order of completion can be derived from this for track analysis).  
- rating scores  
- actions on resources and services (within an action).  
Furthermore the portfolio also contains identity information and personal profiles. The actor can make selections of this information available for others to view (and so creating multiple identities).  
Import of portfolio data will be supported later.  
Historical data can also be entered manually in the portfolio view (e.g. previous acquired competence levels and products and diplomas/certificates (as a proof for the proficiency levels). |
| function/job in domain      | A domain of knowledge can contain different actors with different functions (in work called job functions/job titles). Functions/jobs are constructs to use in competence maps to provide for a natural way to specify different types of competence maps, especially when a domain contains different
job functions at different levels.

A functions/job in the domain can be the focus of learning. E.g. in the domain medical profession there are functions/job titles like doctors, nurses, etc.

A functions/job has a set of competences defined that define the minimum requirements for that function.

The competences that are specified under each function specify the target-proficiency levels of that competence that must be met in order to obtain the function (at the specific level of the function).

In a Competence Map at least one function/job should be defined.

Function/jobs can be performed at one or more function/job levels, like Trainer, Master, and Trainee.

In science there are undergraduate students, PhD students, postdocs and professors.

An actor can have several goals to use the system. These goals can be specified in the system so that the system can help the user to fulfill his specific requirements (e.g. by a wizard).

An actor specifies his/her goal in the system. This goal is of any of the following types:

a. Search a resource/activity/UOL (‘want to know something’)
b. Search for a route (study a new function/job level)
c. Keep up-to-date within an existing function/job level
d. Improve a proficiency level of a competence
e. Browse the learning network (meta goal to define a goal)
f. Want to reflect on my competences (meta goal to define a goal)
g. Want some support

The optional ‘description’ of a goal is used when the goal itself must be made explicit (the actor formulates his/her goal in a text box).

Examples of concrete goals are:

ad a) Want to know how to tune the guitar
ad b) Want to attain a masters degree in psychology
ad c) I am an optician: what are the current trends, new competences, new knowledge required, etc.
ad d) Want to improve my UML modelling competence
ad e) I want to explore the learning network about psychology to get an overview of the domain
ad f) Which function/job levels do fit my current competence levels? How does my competence levels compare to the others in the learning network?
ad g) Is there anybody available who can help me with ... (in the context of an action)

The goal of the actor occurs in the relationship between the actor and the ecological niche (a problem occurs, a goal is set).

**Action Requests:**

When a person has specifies his goal, this leads to an action request. Two situations can occur:

Start searching for an appropriate action within the current network that fits the goal.

When an adequate action exists the person actor can perform this action.

When no adequate action exist, the action request itself will hold the request. These requests can be read by another actor who can author an adequate action to the system (e.g. add a learning activity).

This mechanism takes care for the social exchange based on user-requests and is governed by social exchange policies.

---

<table>
<thead>
<tr>
<th><strong>Class Name</strong></th>
<th><strong>Documentation</strong></th>
</tr>
</thead>
</table>
| IMS LD         | IMS LD level A, B and C is used as a standard to import/export and exchange Units of Learning (and Units of Assessment). IMS LD uses different other specs like:  
|                | - IMS Content Packaging  
|                | - IMS/IEEE LOM or any other metadata schema  
|                | - IMS QTI for the inclusion of test activities  
|                | - IMS Simple Sequencing (rare case, will not be supported in early releases) |
| IMS QTI        | Assessment activities (items and collections of items) can be exported to (and imported from) the IMS QTI format (latest version). |
| knowledge resource | Knowledge resources are any kind of resources that can be used in learning. Typical resources are:  
|                  | a. HTML pages  
<p>|                  | b. Podcasts / Vodcasts |</p>
<table>
<thead>
<tr>
<th>Class Name</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>c. digital documents</td>
<td>A knowledge resources have a URL as identifier. Knowledge resources can be searched at the level of the learning network or by browsing the topics. Knowledge resources can be grouped using specifications like SCORM.</td>
</tr>
<tr>
<td>d. computer programmes</td>
<td></td>
</tr>
<tr>
<td>e. ...</td>
<td></td>
</tr>
<tr>
<td>learning activity</td>
<td>Learning activities are tasks for learners that describe what they are advised to do in order to attain certain learning or assessment objectives, given some prerequisites.</td>
</tr>
<tr>
<td>learning network</td>
<td>A learning network is an ensemble of actors, learning resources (actions &amp; knowledge resources) and competence maps which are mutually connected through and supported by information and communication technologies to support lifelong competence development. A learning network is defined on a domain that represents the profession or the domain of knowledge. This domain is also used as the default value for the Competence Map. The creator of a Learning Network is also the ‘owner’, i.e. the one who has admin rights on the learning network and can set rights for others (also admin rights). At a higher level than the learning networks it is expected that there is a discovery service to find, subscribe and rate adequate learning networks.</td>
</tr>
<tr>
<td>learning path spec</td>
<td>This spec will be developed as part of the TENCompetence activities in WP7. It must include a scheduling specification for the elements in the route, including information about its availability (e.g. face to face settings, or a minimum number of students to be able to start in a collaborative course).</td>
</tr>
<tr>
<td>process log</td>
<td>The process log contains all digital available data that are produced during the performance of an action, e.g. what did a person access, in what order, how long, who did he/she contact, how often did he logon to the system, etc. These data needs to be designed. All or parts of these data will be stored in the portfolio and the data are available for the competence assessment procedure. The process logs in the portfolio provide the basis means for</td>
</tr>
</tbody>
</table>

TENCompetence – IST-2005-027087 Page 112 / 146
<table>
<thead>
<tr>
<th><strong>Class Name</strong></th>
<th><strong>Documentation</strong></th>
</tr>
</thead>
</table>
| product        | The products that are produced by the actor during the performance of an action (in fact, these products are added as knowledge resources to the learning network). Examples are:  
- reports  
- mail messages  
- wiki contributions  
- blog texts  
- audio/video recordings  
- computer programmes developed  
- etc. |
| proficiency level | A competence has 1 or more proficiency levels that are an integral part of the competence itself.  
The proficiency level can be any range of Integer numbers and indicates the minimum level that needs to be attained for the attainment of a certain function/job in the domain (e.g. being a proficient postdoc researcher in psychology).  
Zero means that the competence is absent. Depending on the community one can decide to make dichotomies (0 is absent, 1 is present), or a number of discrete levels (e.g. in a psychology curriculum 12 levels of proficiency are distinguished for the diagnosis competence: 0-11).  
For each role a set of competences is defined together with the target proficiency-level that must be met in order to meet the requirements for the function/job level. E.g.:  
Function/job level A requires:  
- Competence A at level 3  
- Competence B at level 4  
- Competence C at level 1  
A proficiency level can have two states: a) valid or b) invalid. This is used for competences that must be updated frequently. For a doctor and a pilot the competences must be updated frequently to keep the proficiency level.  
When a proficiency level is invalid a new competence assessment has to take place to make it valid again. By default a proficiency level is valid and it can only change to invalid when the attribute 'period of validity' has set to a value (it is an optional attribute; when it has no value the proficiency level is
always valid). The proficiency level is invalid when
the current date > registration date of the proficiency level +
period of validity

The registration date of the proficiency level must be registered
in the actors portfolio. It is the date that the competence
assessment has set the competence level for the actor.

Example:

For instance, pilots must have at least a minimum number of
flying hours to keep the validity of the proficiency level on the
flying competence. Doctors must have at least a number of
training points per year in order to keep their certificate (=role).

How this is modelled is as follows:

Job level = medical doctor in child care
Competence= ´state-of-the-art knowledge of the field´
Proficiency level = 1 (period of validity = 12 month)

After 12 month the proficiency level of the competence will be
invalid. This means that the role of doctor is also invalid at that
moment. A new competence assessment is needed to turn it to
valid again. In this competence assessment one can look
whether the doctor has attained the required number of points
(looking at the results).

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>always valid). The proficiency level is invalid when the current date &gt; registration date of the proficiency level + period of validity</td>
</tr>
<tr>
<td></td>
<td>The registration date of the proficiency level must be registered in the actors portfolio. It is the date that the competence assessment has set the competence level for the actor.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>For instance, pilots must have at least a minimum number of flying hours to keep the validity of the proficiency level on the flying competence. Doctors must have at least a number of training points per year in order to keep their certificate (=role). How this is modelled is as follows:</td>
</tr>
<tr>
<td></td>
<td>Job level = medical doctor in child care</td>
</tr>
<tr>
<td></td>
<td>Competence= ´state-of-the-art knowledge of the field´</td>
</tr>
<tr>
<td></td>
<td>Proficiency level = 1 (period of validity = 12 month)</td>
</tr>
<tr>
<td></td>
<td>After 12 month the proficiency level of the competence will be invalid. This means that the role of doctor is also invalid at that moment. A new competence assessment is needed to turn it to valid again. In this competence assessment one can look whether the doctor has attained the required number of points (looking at the results).</td>
</tr>
<tr>
<td>result</td>
<td>Results of actions are:</td>
</tr>
<tr>
<td></td>
<td>a. All types of products that are produced during learning. These products can be large (a report, thesis, etc) or very small (an email, forum contribution, wiki contribution, etc.).</td>
</tr>
<tr>
<td></td>
<td>b. Process logs. This contains a logging off all the logged actions of the user, including the time stamps.</td>
</tr>
<tr>
<td></td>
<td>c. Assessment results. All results of all assessments are recorded together with the dates started/attained, the results and the possible links to products and/or logs.</td>
</tr>
<tr>
<td></td>
<td>Results are used as ´evidence´ for the competence assessment to decide which proficiency level a person has on several competences (and as a result on the function/job levels).</td>
</tr>
<tr>
<td></td>
<td>For a result it can be recorded in the attribute ´completed´ whether the action was completed (true or false).</td>
</tr>
<tr>
<td>RSS</td>
<td>An actor can use RSS feeds from the learning network. Feeds can include:</td>
</tr>
<tr>
<td></td>
<td>a. New Actions in a learning network</td>
</tr>
<tr>
<td></td>
<td>b. New knowledge resources (e.g. podcasts, vodcasts, blogs,)</td>
</tr>
<tr>
<td>Class Name</td>
<td>Documentation</td>
</tr>
<tr>
<td>------------</td>
<td>---------------</td>
</tr>
</tbody>
</table>
| digital files) | c. New Competences/proficiency levels  
d. New Roles  
e. Requests from other actors with pending goals  
Required is minimal RSS 2.* for v/podcasts. Possibly also Atom feeds can be supported. |
| schedule | Actions can be scheduled by an actor.  
The design of the schedule is rather complex and has to be elaborated separately. At this moment it is represented in the domain model as a single class to indicate that the schedule is available. |
| support activity | Support activities are activities of an actor to help or support other actors, e.g. provide help, tutoring activities, advise. |
| topic | Actions (including resources, services) and Competences can be organised in ‘topics’. E.g. in the domain psychology, the knowledge resources can be organised into topics like:  
a. Methodology  
b. Clinical psychology  
c. Educational psychology  
d. etc.  
The topics can be used to browse the actions in the network from a topic view instead of a competence view.  
For first implementations, the topics are nothing more than one or more keywords that can be added to (the metadata of) each action. The system should help the user to type in a key word that already exists. When the user types a new keyword it will be added to the list of keywords.  
The keywords can be viewed alphabetically (maybe also on ‘number of times used’) and people can use the keywords to filter actions and knowledge resources.  
This topic structure can later be quite advanced (e.g. structured as an ontology or as a topic map). As a standard ISO Topic Maps or RDF can be used to import or export them.  
Just like competence maps: a topic cannot be shared between networks because of authorisation issues.  
In this diagram, the topic structure is only represented with one class. Depending on the implementation this topic structure should be more elaborated. |
### Class Name Documentation

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>unit of assessment</td>
<td>A Unit of Assessment (UOA) is a part of the new Assessment Spec that will be developed in WP6 of TENCompetence. At the moment it is: a. A specialised unit of learning that is responsible for the processing of the workflow behind more advanced assessments. A single test is represented by an assessment activity. The UOA can bundle various test activities, learning activities and support activities and can put them in a workflow to represent new assessment types like peer assessment and 360 degrees feedback. Units of Learning (possibly also with the inclusion of one or more assessment activities) can be combined with a UOA by using a route, e.g. a sequence of [UOL; UOA] or a selection of [UOL; UOA]. In a sequence the UOL (e.g. a course) is done before the assessment. In a selection the UOL and the assessment are done in parallel. This same mechanism can also be used to support the competence assessment. To summarize: * QTI tests are used to model classical tests in the form of an assessment activity. * Assessment activities can be used separately or in a route in combination with other actions. * Assessment activities can be used in the context of a UOL (the mechanisms proposed in IMS to include IMS LD with IMS QTI) * Units of Assessments are specialised Units of Learning that combine a series of assessment activities, learning activities and support activities with a multi-role workflow to model advanced, workflow based tests like peer assessment. * Units of Assessment can be combined with Units of Learning using routes * Units of Assessment will also be used for competence assessment.</td>
</tr>
<tr>
<td>unit of learning (UOL)</td>
<td>A unit of learning is a plan for a learning and/or teaching process that could be performed to attain certain learning objectives, given certain prerequisites. They are modelled in IMS Learning Design (XML file that conforms to the spec; authored with IMS LD tools). This will be included when WP6 work is finished. Examples of units of learning are: course plans, lesson plans, workshop plans, etc. A unit of learning contains (references to):</td>
</tr>
<tr>
<td>Class Name</td>
<td>Documentation</td>
</tr>
<tr>
<td>------------</td>
<td>---------------</td>
</tr>
<tr>
<td></td>
<td>- learning objectives and prerequisites</td>
</tr>
<tr>
<td></td>
<td>- learning activities - support activities</td>
</tr>
<tr>
<td></td>
<td>- a description of the roles that perform activities in the unit of learning</td>
</tr>
<tr>
<td></td>
<td>- a description of the learning objects and services that are needed during the execution of the activities</td>
</tr>
<tr>
<td></td>
<td>- a description of the information that should be stored during the unit of learning, including portfolio information</td>
</tr>
<tr>
<td></td>
<td>- a description of the teaching-learning process (learning design): which role performs which activities in which sequence?</td>
</tr>
<tr>
<td></td>
<td>- a description of personalisation/adaptation aspects (certain persons get different resources/activities depending on their profile) Units of Learning are formally modelled and interchanged using the IMS Learning Design specification.</td>
</tr>
<tr>
<td></td>
<td>Units of learning can contain time information that is related to the schedule.</td>
</tr>
</tbody>
</table>
9.2 **Appendix 2 – Initial Requirements Report (Version 2.2)**

**Introduction**

This document describes the high level functional requirements for the TENCompetence infrastructure. These requirements define the functions of the TENCompetence infrastructure from a users’ perspective. The functional requirements are defined as use cases that describe common problems of lifelong learning.

This document is structured in two sections. The first section provides general definitions for the concepts and external systems to which the use cases refer. The second section describes the functional requirements as seven high level use cases.

## 1. Definitions

### 1.1. Acronyms

The following acronyms are used in the rest of this document.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDP</td>
<td>Competence Development Programme</td>
</tr>
<tr>
<td>TENCC</td>
<td>TENCompetence Client</td>
</tr>
<tr>
<td>UoL</td>
<td>Unit of Learning</td>
</tr>
</tbody>
</table>

### 1.2. Actors

Learning and training are actor centric tasks, to which actors contribute in different roles. These roles are not fixed but these roles can change, overlap and interfere from the perspective of a single actor. These dynamics of roles play an important part in lifelong competence development. Therefore, a support system for lifelong competence development has to estimate and to reflect the support of the needs of the different roles, in order to provide support throughout these tasks.

From the perspective of the technological infrastructure, the actors trigger the system’s functions from the outside. With regard to the TENCompetence infrastructure, actors can be persons, groups, a formal or informal organisation, or even other systems (e.g. knowledge repositories, or autonomous agents). Each actor can interact with the system from within different roles. These roles may change according to policies of a learning network, but also according to the choice of the actor. Thus, an actor can act as a learner, a teacher, an assessor. The roles of an actor may vary according to the learning network, to the competence levels, to job positions, etc. Actors may also perform in different roles simultaneously.

For the TENCompetence project there were six user-stereotypes identified. The stereotypes are actors, assessors, authors, external systems, (learning) facilitators and learners (see Figure 1). Each user-stereotype clusters roles of similar function in a learning network. The use cases of the functional requirements are always associated to one or more of these user-stereotypes.
Figure 1: User-stereotypes

**Actors**

The most general user-stereotype is the *actor*-stereotype. This stereotype refers to any type of external interaction without reflecting a specific role. In the requirements specification of TENCompetence the actor stereotype is only used to specify use cases that can not be associated reasonably to one of the other user-stereotypes.

**Learners**

With regard to the high level use cases, the learner stereotype is most important. The *learner*-stereotype represents those persons who use the TENCompetence infrastructure in order to develop their competences. This can be a pre-defined educational path provided such as a university, a competence development programme in vocational training within an enterprise, or participation in a community of practice. The learning activities within the TENCompetence infrastructure can be part of both formal and informal learning simultaneously.

**Authors**

The *author*-stereotype refers to those actors who create knowledge resources, activity specifications or entire curricula. In general terms all editing actions within the TENCompetence infrastructure are associated to the author-stereotype. This covers at least the following roles.

1. Content creators develop knowledge resources for a learning network.
2. Instructional designers or course designers define topics and create the actions of parts of a learning network.
3. Administrators specify and associate competence maps to a learning network for assessment.
Facilitators
The facilitator-stereotype refers to those roles that actively support learners during their learning process. The users of the TENCompetence infrastructure act as learning facilitators when they support learners as experts, teachers and trainers, tutors, mentors, moderators, or peer learners.

Assessors
An assessor is a person that is legally entitled for accreditation of the learning outcomes of a learner. An assessor can be involved in the competence assessment, an assessment activity or the unit of assessment. The role of assessor can be fulfilled by any actor (also the person himself, e.g. for self-assessment) dependent on how formal the assessment must be.

External Systems
A special case is the external-system-stereotype. This particular user-stereotype refers to external systems that use the TENCompetence environment or provide information to it. These systems offer special services on the level of assessors, authors, or facilitators. This user-stereotype helps to define the boundaries of the TENCompetence environment’s functionality. The TENCompetence environment and the external services form the TENCompetence infrastructure.

The user-stereotypes refer to specific types of user activities or intentions. From the perspective of the real world a user may change its role with regard to these stereotypes. This is illustrated by the following example. Teachers and Tutors use the system to help Learners in their learning process. By these means their primary role while using the system refers to the facilitator-stereotype. Different to tutors, teachers are also involved in the preparation of units of learning and in testing the learning progress of the students. During preparation, teachers act as authors to the system, while during testing teachers act as assessors. Compared to teachers, tutors have fewer responsibilities in educational processes, although they may add resources to a course or help during assessment tasks. However, the described user-stereotypes are not the same as professional roles but refer to different ways of using the TENCompetence infrastructure.

1.3. Main research activities
The main TENCompetence research activities focus on four complexity levels of technology enhanced learning. These levels of complexity refer to knowledge resources, units of learning and learning activities, competence development programmes, and learning networks. On each level specific research and development tasks are conducted within the TENCompetence project. These research activities lead to services and components that will be integrated into the TENCompetence infrastructure. The initial requirements on each of these levels refer to the anticipated results of the research and development activities.

1.3.1. Knowledge resources
Knowledge resources are any kind of resources that can be used in learning and teaching. Typical resources are text documents, such as HTML pages, audio, video, or computer programs, such as simulations. All knowledge resources that should be used with the TENCompetence infrastructure require a unified resource identifier (URI). If resources from a web-based knowledge resource repository are used, a unified
resource locator (URL) to the resource in the repository is a sufficient identifier of a resource. They can be searched at the level of a learning network or by browsing the topics of a unit of learning. Knowledge resources can get grouped by using specifications such as SCORM or IMS Content Packaging.

1.3.2. Units of Learning and Learning Activities
Learning activities are the designed or performed activities of a person that are directed at the attainment of a (explicit or implicit) learning objective. Designed learning activities are called 'units of learning' (UoL), such as courses, workshops, lessons, etc. A unit of learning adds a 'learning design' to the knowledge resources; they add pedagogical aids like study tasks, tutoring, mentoring, monitoring communication services, feedback, formative and summative assessments. Units of Learning are formally modelled and interchanged using the IMS Learning Design specification.

1.3.3. Competence Development Programmes
We define Competence Development Programmes (CDP; synonyms: route, learning path, curriculum, programme) as formal, non-formal, or informal collections of learning activities and UoLs, which are used to build competence in a certain discipline or job. Within a CDP the learning activities and units of learning are loosely coupled whereas in a UoL, learning activities are tightly integrated. Compared to UoLs are CDPs a more flexible approach for organising learning processes. Depending on the competences to be built, these programmes can be small or quite extensive.

1.3.4. Learning Networks
A learning network is an ensemble of actors, knowledge resources, units of learning, competence development plans, and competence maps which are mutually connected through and supported by information and communication technologies to support lifelong competence development. This characterisation, however, fails to explain how learning networks may contribute to the effectiveness, efficiency or even attractiveness of learning. The concepts of learning networks refer to the social ties between actors, their roles in the learning community, and their learning objectives.

1.4. External Services
The high level requirements refer to several external services that are provided by external systems. The external-system stereotype is used to refer to those systems or system components that provide the described service. Those services that are mentioned in the high-level use-case are at the core of the TENCompetence infrastructure and are partially shared across the project’s research activities. The external services exist independently from the TENCompetence’s core system. The core system has to provide interfaces to exchange with those systems that provide the external services.

1.4.1. Portfolio service
The portfolio service collects and organises user data including identity information and personal profiles. The service provides different views on this data for different purposes. These purposes can be self-reflection, assessment of learning progress, publishing a profile (e.g. a CV) to a specific group (e.g. an employer). These views
can be printed in a standard format such as the EuroPass CV; or exported in a standard structure like IMS ePortfolio in order to be used by another ePortfolio system. An actor can select which of this information is private or available to others. This includes that an actor can maintain multiple “identities”.

With regard to TENCompetence the portfolio service can contain all relevant information of an actors learning process. This includes information about past learning experiences, and competences, diplomas, or certificates that were acquired outside of the system. Within the TENCompetence infrastructure, the portfolio service contains data about all actions of an actor. This data includes – but is not limited to – stopped but incomplete actions, completed actions, results of learning activities, assessment results, rating scores, and actions performed on resources or external services.

1.4.2. Positioning service

Positioning refers to the mapping of learner characteristics to CDPs. The learner characteristics are provided in an ePortfolio format. The positioning service uses the registered actions-results that are found in the portfolio to estimate an actor’s initial proficiency levels of each competence defined in the competence map of a CDP. For instance, this is achieved if persons enter a learning network and provide their (non-empty) ePortfolios. This informs these persons about their relation to the competence map of the CDP. The results of the positioning can be further used for navigational purposes.

The positioning service can also be used by the TENCompetence environment for the interim assessment of competence levels of a learner. This assessment is based on the analysis of the learner’s actions in a learning network which are collated in an ePortfolio. Furthermore, a positioning service may provide information about estimated competence levels of similar or related competences even if they are not directly related to the previous actions.

1.4.3. Navigation Service

A navigation service guides learners through a learning network. It is designed to identify suitable learning activities, UoLs or CDPs according to the learners’ competence profile. For this purpose the service takes learners’ profiles and the behaviour of other learners into account.

Within the TENCompetence infrastructure a navigation service is designed as a personalised recommender system. The recommendations are based on the learners’ profiles and their relation to the subscribed learning network. As a personalised recommender system it takes the previous actions of learners with similar profiles into account for recommending automatically generated CDPs.

1.4.4. Notification service

The notification service helps the actors in a learning network to stay informed about various events. This service handles the distribution of newsletters, text messages, deadline reminders, etc. This service may notify the actors through different channels, such as e-mail, SMS, or instant messages.

As part of the TENCompetence infrastructure, this service can select appropriate notifications according to the context of a learner in a learning network. This may include that the service sends an SMS instead of an e-mail if the user is not using the
systems from a desktop computer or laptop; or the service notifies learners about learning opportunities based on their location.

2. **Functional requirements**

This part describes six high level use cases that describe the main users’ motivation for accessing TENCompetence. The use cases describe the functions of the system from a user perspective. Each high level use case contains a set of sub-ordinate use cases that describe the functions of the system that a user wants to use.

The term “functional requirements” refers to the description of functions a system must provide to its users. These requirements can be used in order to validate if a system provides a specified function. Functional requirements define the users’ objectives for a system, independent from the actual implementation of that system. Therefore, functional requirements are not necessarily strict definitions that lead only to one implementation. For example, the same functional requirements can be met by a rich client and a web-based environment even if these systems neither base on the same code base nor have similar user-interfaces.

The use cases are defined as six key tasks for which the TENCompetence infrastructure will be used in the future. Each use case describes a distinct set of functions that are needed for lifelong competence development. The use cases are documented according to the following list.

- Explore a learning network
- Improve the proficiency level
- Keep up to date
- Reflect on their competences
- Study for a new function or a new job
- Want some support for some specific action

The high level use cases have sub-ordinate task assigned. Each of these use cases is described as a narrative as well as a UML use case model. Both, the narrative and the UML model describe the functional tasks in order to achieve the goal of the use case.

Each use case description has the following parts.

1. The general objective for the use case from the perspective of the main user
2. A UML diagram of the use case
3. All actors that are directly involved in the use case
4. The prerequisites that have to be met for the given use case
5. A narrative description of the UML model

### 2.1. **Explore a Learning Network**

**2.1.1. Objective of the use case**

Learners need to explore a learning network in order to find topics, actions, etc., that suit their aspirations, and learning needs.
2.1.2. UML use case diagram

2.1.3. Involved user-stereotypes
- learner (main user)

2.1.4. Pre-requisites
- The learner must be connected to the system
- The learner has to be registered to the system
- The learner has to be authenticated by the system

2.1.5. Narrative description

Select learning network to explore

Every time learners log into the system, they receive the list of learning networks to which they are subscribed. Learners may also return to the list of subscribed learning networks at any time they use the system. Additionally, a learner can select issues and topics of interest. Based on this selection the learner receives a list of relevant learning networks. The system can also propose some topics according the current context or previous behaviour of the learner. Once a learning network of interest has been identified, a learner can select this learning network in order to explore it.

Explore learning network

After a learner has selected a learning network, the learning network can be explored. Exploring means browsing or searching the contents and members of the learning network. While a learner is exploring a learning network, the system has to provide information about activities, UoLs, CDPs, competence maps, and people. An exploring learner will not directly subscribe to a learning activity, UoL, CDP, or competence map.
The system has to provide opportunities to explore a learning network
- with regard to the related learning activities and UoLs (Browse or search list of learning activities and UoLs);
- with regard to the knowledge resources of the learning network (Browse or search list of resources);
- with regard to the competence maps that are covered by the learning network (Browse or search competence maps); and
- with regard to the members of the learning network as well as their profiles (Browse or search people)

Subscribe to action
Once a learner has enough knowledge about a specific learning network and the possible actions within the network, the learner can choose to subscribe to a learning activity, a UoL, or a CDP that is available in the selected learning network.

2.2. Improve Proficiency Level

2.2.1. Objective of the use case
Learners need to improve their proficiency level for a specific competence or need to acquire new competences.

UML use case diagram
2.2.2. Involved user-stereotypes:
- learner (main user)
- portfolio system (external system)
- positioning service (external system)

2.2.3. Pre-requisites
- The learner must be connected to the system
- The learner has to be registered to the system
- The learner has to be authenticated by the system
- The learner has subscribed to a learning network
- The learner has an ePortfolio in a portfolio system that is capable to handle that contains competence related information; if a learner does not have a ePortfolio, the TENCompetence system may create a new ePortfolio for the learner.

2.2.4. Narrative description

(re-)estimate competence levels

When learners look for new learning opportunities one option is to filter the suitable learning activities of a learning network with respect to the learners’ proficiency levels. A learner can choose “estimate proficiency level” whenever the ePortfolio has changed or no information about proficiency levels can be found in the learner’s ePortfolio.

In order to limit the proficiency level estimation a learner can select topics, knowledge resources, learning activities, or UoLs of the selected learning network.

Provide competence levels for the learner

A learner’s competence levels are estimated with regard to the entire learning network the learner has subscribed and to the information that is stored in the ePortfolio. If an ePortfolio contains competence information, this information contains the name of the competence and its proficiency level.

The learner’s ePortfolio is stored in an external portfolio service. The system has to request the ePortfolio from the external service.

If the ePortfolio contains no information about competence levels or the competence levels found in the ePortfolio were not updated after the last change of the ePortfolio, the system has to forward the ePortfolio to an external positioning service in order to estimate the learner’s proficiency levels.

The positioning service analyses the explicit and implicit competence information found in the learner’s ePortfolio with regard to the competences that are covered by the learning network.

The information related to the learning network can be limited by the service according to the learner’s selection of topics, knowledge resources, learning activities, or UoLs.

This positioning service returns the competences that are found in a learner’s ePortfolio according to the learning network. Each item of this set contains a competence name and an estimated proficiency level.
The system forwards the result from the *positioning service* to the *portfolio service* in order to update the learner’s ePortfolio. Additionally, *learners* can manually determine their competence levels, by changing the levels that are in the ePortfolio.

**Select competence map**

The system uses the competence levels that are available in the learner’s ePortfolio to select those competence maps of the learning network that are suitable to improve the learner’s proficiency levels. The system has to identify those competence maps that
- improve the learner’s existing competences, or
- offer new competences according to the learner’s ePortfolio.

The learner can select one competence map the learner is interested in.

**Select ‘improve proficiency level’**

Each competence map has several associated competences. The competence map contains also information from which proficiency level to which proficiency level these competences can be developed. According to the learner’s ePortfolio, the system should present those competences of the competence map that can be developed. Competences that require a higher entry proficiency level than the current proficiency level of the learner should be hidden. The system should also indicate the required effort of developing a competence. The estimate for the effort is the difference between the current proficiency level of the learner and the final proficiency level of a competence that is defined by the competence map.

The learner can select one competence that should be improved by selecting “improve proficiency level”.

**Search alternative actions**

Based on the selected competence the system should search for suitable
- learning activities (search suitable learning activities),
- UoLs (search suitable UoLs), and
- CDPs (search suitable CDP).

The result has to contain only those learning opportunities that meet the current learning needs for improving the learner’s proficiency level to the next level. The resulting learning opportunities are of equivalent with regard to the competence that should be developed.

**Select or subscribe to action**

The learner selects one of the offered learning opportunities in order to start the development of the competence. If the selected learning activity, UoL, CDP requires subscription, the system should subscribe the learner to it.
2.3. Keep up to date

2.3.1. Objective of the use case
Knowledge, skills, etc. change over time and professionals must regularly update their competences in order to maintain their proficiency level for a specific function or job.

UML use case diagram

2.3.2. Involved user-stereotypes
- learner (main user)
- portfolio system (External System)
- positioning service (External System)
- notification service (External System)

2.3.3. Pre-requisites
- The learner must be connected to the system
- The learner has to be registered to the system
- The learner has to be authenticated by the system
- The learner has subscribed to a learning network
- The learner has an ePortfolio that contains competence related information
- The learner holds a function or has a job with a known competence map

### 2.3.4. Narrative description

***(re-)estimate competence levels***

When learners look for learning opportunities related to their current function or job, one option is to filter the suitable learning activities of a learning network with respect to the learners’ proficiency levels. A learner can choose “estimate proficiency level” whenever the ePortfolio has changed or no information about proficiency levels can be found in the learner’s ePortfolio.

*The competence map of the function or job is used to limit the proficiency level estimation for the selected learning network.*

**Provide information about current function or job and competence levels**

A learner’s competence levels are estimated with regard to the entire learning network the learner has subscribed, and to the information that is stored in the ePortfolio. If an ePortfolio contains competence information, this information contains the name of the competence and its proficiency level.

The learner’s ePortfolio is stored in an external portfolio service. The system has to request the ePortfolio from the external service.

If the ePortfolio contains no information about competence levels or the competence levels found in the ePortfolio were not updated after the last change of the ePortfolio, the system has to forward the ePortfolio to an external positioning service in order to estimate the learner’s proficiency levels.

*The positioning service* analyses the explicit and implicit competence information found in the learner’s ePortfolio with regard to the competences that are covered by the learning network.

*The information related to the learning network is limited by the service according to the competence map of the learner’s current function or job.*

This positioning service returns a set of the competences that are found in a learner’s ePortfolio according to the learning network. Each item of this set contains a competence name and an estimated proficiency level.

The system forwards the result from the positioning service to the portfolio service in order to update the learner’s ePortfolio.

Additionally, learners can manually determine their competence levels, by changing the levels that are in the ePortfolio.

**Select “keep up-to-date”**

Each function or job has a competence map assigned to it. The competence map contains also information about the competences and proficiency levels that are required for the function or job.

By selecting “keep up-to-date” the learners match their competences and proficiency levels with those of the current function or job.
Search for competence updates

The learner can search the learning network for competence maps. The search is filtered by the learner’s competence information in the ePortfolio, the competence map for the function or job the learner holds, and the available competence maps of the learning network. There can be two types of this search.

1. **Search for new competences of the current function or job position.**
2. **Search for higher competence levels than the learner’s current that are required for the function or job position.**

The search should list those competence maps that are of relevance for the learner’s development to fit the developments of the function or job. The competence maps in this list must contain competences with levels that are higher than the learner’s current level for the related competence of the competence map.

Provide automatic updates on a schedule via e-mail

A learner can select a competence map in order to receive updates on changes of those learning activities, UoLs, or CDPs that are related to the selected competence map. An external notification service will inform the learner about changes – such as schedules or new dates – of the learning activities, UoLs, and CDPs bound to the selected competence map.

Search for new learning activities or UoLs for current function or job

The learner can choose to search for new learning activities, UoLs, and CDPs for the current function or job. The search filters those learning activities, UoLs and CDPs of the learning network that lead at least to the same competence levels the learner holds for the current function or job position. Again, these competence levels are reflected in the learner’s ePortfolio.

Search alternative actions

Based on the selected competence the system should search for suitable

- learning activities (search suitable learning activities),
- UoLs (search suitable UoLs), and
- CDPs (search suitable CDP).

The result has to contain only those learning opportunities that meet the current learning needs for improving the learner’s proficiency level to the next level. The resulting learning opportunities are of equivalent with regard to the competence that should be developed.

Select or subscribe to action

The learner selects one of the offered learning opportunities in order to start the development of the competence. If the selected learning activity, UoL, CDP requires subscription, the system should subscribe the learner to it.
2.4. Reflect on competences

2.4.1. Objective of the use case
Based on the learner’s current proficiency levels found in the ePortfolio, the system will show all the new functions or jobs descriptions that (largely) match with these competence levels.

2.4.2. UML use case diagram

2.4.3. Involved user-stereotypes
- learner (main user)
- portfolio system (External System)
- positioning service (External System)

2.4.4. Pre-requisites
- The learner must be connected to the system
- The learner has to be registered to the system
- The learner has to be authenticated by the system
- The learner has subscribed to a learning network
- The learner has an ePortfolio that contains competence related information
Narrative description

(re-)estimate competence levels

When learners look for new learning opportunities one option is to filter the suitable learning activities of a learning network with respect to the learners’ proficiency levels. A learner can choose “estimate proficiency level” whenever the ePortfolio has changed or no information about proficiency levels can be found in the learner’s ePortfolio.

In order to limit the proficiency level estimation a learner can select topics, knowledge resources, learning activities, or UoLs of the selected learning network.

Provide competence levels for the learner

A learner’s competence levels are estimated with regard to the entire learning network the learner has subscribed and to the information that is stored in the ePortfolio. If an ePortfolio contains competence information, this information contains the name of the competence and its proficiency level.

The learner’s ePortfolio is stored in an external portfolio service. The system has to request the ePortfolio from the external service.

If the ePortfolio contains no information about competence levels or the competence levels found in the ePortfolio were not updated after the last change of the ePortfolio, the system has to forward the ePortfolio to an external positioning service in order to estimate the learner’s proficiency levels.

The positioning service analyses the explicit and implicit competence information found in the learner’s ePortfolio with regard to the competences that are covered by the learning network.

The information related to the learning network can be limited by the service according to the learner’s selection of topics, knowledge resources, learning activities, or UoLs.

This positioning service returns a set of the competences that are found in a learner’s ePortfolio according to the learning network. Each item of this set contains a competence name and an estimated proficiency level.

The system forwards the result from the positioning service to the portfolio service in order to update the learner’s ePortfolio.

Additionally, learners can manually determine their competence levels, by changing the levels that are in the ePortfolio.

Select competence map

The learner selects on competence map of the learning network.

The system will show the learner how this competence map matches to the learner’s competences.

According to the selected competence map the system selects all related function or job descriptions.

Reflect on information

The learner reflects on the provided information.

To support the learner during the reflection process the system will provide previous reflection reports by other learners and by the learner, on the given competence map.
Read previous reflection reports
The learner reads reflection reports of other learners or own reflections that have been made previously on the competence map.

Write reflection report
The learner chooses to write a reflection report to record the results of the reflection about the own competences. This reflection report is an informal description of the reflection outcomes. For example it look as following. “Because I want to be a Math teacher later, I will have to work on my didactic skills. I’m first focusing on the mathematical courses, but I’m writing it down so I will think about it later. And so I can show my boss what I’m up to with my Personal Development Plan”.
The reflection reports have to get stored in order to provide a reflection history to the learner.

Ask for function or job assessment
The learner can request an assessment of the learner’s competences with regard to the selected competence map.

Perform function or job assessment
An external positioning service compares the learner’s competences with the competences described in the competence map. The result of this comparison is a competence map that indicates functions or jobs that suit the learner’s current competences.

View resulting set of functions or jobs that fit to the learner
The result of the function or job assessment is provided to the learner. The system indicates which competence maps are related to the applicable functions or jobs.

2.5. Study for a new function or a new job

2.5.1. Objective of the use case
In order to develop competences and skills for a new job or a new position, learners need to find or create appropriate competence development programmes (CDPs) for developing the necessary competences and skills.
2.5.2. UML use case diagram

2.5.3. Involved user-stereotypes

- *learner* (main user)
- *author*
- *navigation service* (External System)

2.5.4. Pre-requisites

- The learner must be connected to the system
- The learner has to be registered to the system
- The learner has to be authenticated by the system
- The learner has subscribed to a learning network
- Competence maps for functions or jobs exist within the learning network

Narrative description

*Select “study for a new function or job”*

The system provides an open search facility that allows *learners* either to pick a function or job description from a list or to define a set of competence criteria that describes the selected function or job best. Learners can express their wish to study for a new job or a function by entering competence criteria and selecting “study new function or job”.

Based on the provided competence criteria the system identifies those CDPs that meet the requested competences.
Select a function or job to study for

The system provides a catalogue-like list of predefined jobs or job functions that are available in a learning network. A learner can select one or more function or job descriptions in which the learner is interested. Each function or job description comes with a set of competence criteria, that can be used for looking for CDPs in the current learning network.

Search for a CDP

The system selects all CDPs that lead to the selected competence criteria. The learners receive a list of CDPs as a response to their request. From this list the learners can choose the best fitting one according to their needs and demands. This list can contain pre-designed CDPs, published successful tracks (good practices) and automatically generated CDPs. The system should indicate the CDPs with regard to the competence maps a learner completely meets, partially meets or does not yet meet.

Make CDP pending

If the selected function or job description has no CDPs assigned to it, a learner can express the personal learning needs by requesting a CDP for the selected competence criteria.

Select and Subscribe to CDP

If a learner identified a suitable CDP, the learner selects this CDP. There are three types of CDPs that are provided to the learners:

- Select pre-designed CDP
  These CDP are provided by one or more authors.

- Select successful track to a function or job
  These CDPs refer to good practices of learning the competences that are required for a function or job. These CDPs are provided by other learners or by authors that monitor the learning network.

- Select an automatically created CDP
  These CDPs are generated automatically by an external navigation service. The service selects learning network’s learning activities or UoLs that are related to the provided competence criteria or from successful learning paths of other learners.

Once a learner has selected a CDP the system has to subscribe the learner to it.

Schedule CDP

After selecting a CDP the learners have to schedule the related learning actions and units of learning (UoL). The system suggests time intervals for the related learning actions and UoLs according to learner’s profile.
Select and subscribe to learning activity

The learner selects one of the offered learning opportunities of the selected CDP in order to start the development of the competence. If the selected learning activity or UoL requires subscription, the system should subscribe the learner to it.

Read pending CDP

Authors have access to the pending CDP requests.

Author CDP

Using the original job or function description provided by a learner for a pending CDP request, an author can create a new CDP or author an existing one. Such a CDP becomes available to the learners as pre-designed CDP. Optionally, the system automatically generates a CDP outline that can be edited and extended by an Author.

Publish successful track towards a function or job

Authors can publish successful tracks for developing the competences for a particular function or job to the learning network. These successful tracks refer to good practices of professional development.

2.6. Want some support

2.6.1. Objective of the use case

Learners need support during an action and after receiving support by a facilitator, learners and facilitators can rate the support activity.
2.6.2. UML use case diagram

2.6.3. Involved user-stereotypes
- learner (main user)
- facilitator
- portfolio service (External System)

2.6.4. Pre-requisites
- The learner must be connected to the system
- The learner has to be registered to the system
- The learner has to be authenticated by the system
- The learner has subscribed to a learning network
- The learner is assigned to a learning activity
- The facilitator must be connected to the system
- The facilitator has to be registered to the system
- The facilitator has to be authenticated by the system
- The facilitator has subscribed to a learning network
2.6.5. Narrative description

Select "want some support"

While performing an action, a learner always has the opportunity to request some support for that action. Each action is assigned to an activity that is assigned to a part of a competence map. Therefore, the support request has the specific context of the competences of the referred part of the competence map as well as of the competence levels of the learner. Within this context, the learner needs the opportunity to specify the support request in greater detail.

Search for a support provider

Based on a support request the system will search for a support provider. This search is restricted by the learner’s context and by a question that has to be entered. The search will return a list of those support providers who are likely to answer the learner’s request.

Register as potential support provider

In order to appear as a support provider in the learning network, facilitators have to register. The registration can be done on the level of:

1. Learning Activities, UoLs or CDP
2. Function/jobs
3. Competences in general (including general competences like “using the TENCC system”)
4. Proficiency level

This limits the types of requests for which an expert is listed. Additionally, facilitators express or provide the areas in which they are able to provide support. E.g. they may provide their competence profile to the system.

Request support from support provider

Based on the list of support providers a learner can directly post a support request to a potential support provider.

Make support question pending

If the learner did not find a suitable support provider, or does not want to address a provider directly, the learner can post the support request to the learning network, this request will be marked as a pending support request.

Read pending support request

All registered support providers of a learning network have access to the pending support requests. The system has to filter the list of pending support requests according their competence and support profile.
Provide support

If a facilitator finds a pending support request which they want and can support, they can choose to provide support to the demanding learner.

Receive support

When a facilitator offers support to a learner, the learner has to receive and accept the provided support.

The system has to inform the learner about the support offers.

Once a learner accepted a support offer, the support request is not labelled as a pending request.

When a learner accepts a support offer from a facilitator the system initiates communication between the learner and the facilitator.

A learner may accept more than one support offer from different facilitators or even join the support offers from multiple facilitators.

Rate and comment the provided support

After learners have received support they have the opportunity to rate and comment the provided support.

Rating refers in these terms to quality indicators. Both sides of the support activity can rate the opposite side. Rating can be done at the level the competences, quality of the support activity, communication style. This kind of rating is a type of ‘peer assessment’.

Commenting is related to the lessons learned from the support action. Commenting is a reflection activity.

Both, ratings and comments are available to the actor to whom it has been assigned. This means that a learner sees the ratings and comments of the facilitator; and the facilitator sees the ratings and comments of the learner.

Rate support aspects

Independently from the learner, the facilitator can rate the competence levels of the support activity as well as the impression of the learner’s competence levels according to the impression of the facilitator.

Store the support activity as action with regard to the function or job of the learner and the facilitator and their competence levels

An external portfolio system will capture the ratings and comments in the portfolios of both participants in the support process. The ratings and comments are stored in the profiles of the learner and of the facilitator by an external portfolio service.

This includes updating both portfolios, the one of the learner and the one of the facilitator.
9.3 Appendix 3 - The TENCompetence Client Scripting Monkey

1.1 Background
The TENCompetence Scripting Monkey is based on the Eclipse Scripting Monkey, an extension to Eclipse that allows scripts to operate on the Eclipse Rich Platform host’s API. Eclipse Monkey is a dynamic scripting tool for the automation of routine programming tasks. Monkey scripts are little JavaScript programs using either the Eclipse APIs or custom Monkey DOMs.

JavaScript is used as the scripting language. Scripts are converted by means of the Rhino library (http://www.mozilla.org/rhino/) to Java calls that call the host’s API.

1.1.1 Screenshot

Figure 37. The Scripting Monkey
1.1.2 Things it can do

The Monkey can do anything that Eclipse or the TENCompetence can do. Monkey see, Monkey do! One caveat, though – it works as long as any function calls are publicly available or provided as a DOM extension (The Monkey Console View is itself a DOM extension). Some possible tasks for the Monkey:

- Search for photos on flickr.com by providing a search phrase. The method returns a bunch of objects containing data about the photos, including a title and enough information to find the actual image itself.
- Automate tasks such as creating batches of users, batch processing of downloads, etc.
- Act as a “Macro”, where many tasks can be strung together consecutively and activated by a key command
- Allow users to add extensions to the client without having to create Eclipse plug-ins in Java
- A script that will search online or local resource files for key words
- A script that can tag multiple resources
- A script that can upload multiple Blog entries to various locations.
- Allow users to add extensions to the client without having to create Eclipse plug-ins in Java
- A script that can upload multiple Blog entries to various locations.
-Invoke Web Services
- …

1.1.3 Usage

The TENCompetence Scripting Monkey is shipped as an Eclipse Plug-in extension. Visually it appears as a View window with menu functions to create and edit new scripts, organise scripts in folders, and run and debug them. There is also a Monkey Console View that displays text output messages and script error messages.

Menu Items:

New Script – Creates a new blank file in the currently selected folder and opens the Text Editor containing the blank script ready for editing.

New Folder – Creates a new folder in the currently selected folder.

Restore Example Scripts – Copies and restores to the Monkey folder all example scripts that ship with the monkey plug-in.

New Folder – Creates a new folder in the currently selected folder.

Edit – Opens the Text Editor containing the currently selected script ready for editing.

Run Script – Runs the currently selected script.

Show Console – Shows the Monkey Script Console View (used for debugging and general text output).
1.1.4 Writing a Script

Scripts are written in JavaScript and follow all the rules of that language. To create a new Script, select the “New Script” menu item from the Toolbar.

Scripts manipulate script objects such as strings and arrays, native Java objects including native Eclipse objects, and special DOM objects made specifically to simplify scripting.

Native Java objects can be accessed through their fully qualified class names. Eclipse objects require that the fully qualified name be prepended with the additional name "Packages". Here are some valid scripting expressions that show how this works:

- `java.lang.System.out.println("hello world");`
- `Packages.org.eclipse.jface.dialogs.MessageDialog.openInformation(window.getShell(), "Monkey Dialog", "Hello World")`

One of the challenges in scripting for Eclipse is dealing with the restricted namespaces of the various plug-ins. The classes and packages that are directly referenceable are those in the plug-in pre-requisites of the org.eclipse.eclipsemonkey plug-in such as org.eclipse.ui and org.eclipse.jface.text. To use other packages and classes, you must use a DOM (see below).

1.1.5 Running Scripts

A Script is run by selecting it in the Monkey Script Viewer and clicking on the “Run Script” button.

Future versions of the Monkey will offer additional ways to run scripts, such as triggering on changes to a files or other resources, or on specific events (e.g., on team synchronize, or on publish war to app server, etc).

1.1.6 Example Scripts

1. Script to display a dialog box with a message:

```javascript
/*
 * Menu: Hello > World
 * Kudos: Phillipus
 * License: EPL 1.0
 */

function main() {
    text = "Hello World from the TenCC Scripting Monkey!\n\n";
    text += "The quick brown fox jumped over the lazy dog's back."
    text += "Now is the time for all good men to come to the aid of their country.";

    Packages.org.eclipse.jface.dialogs.MessageDialog.openInformation(
        window.getShell(),
        "TenCC Monkey Dialog",
        text);
}
```
2. Script to display a dialog box to create a User in the TENCompetence Client:

```javascript
function main() {
  /*
  * Menu: Dialogs > Conduits
  * Kudos: Phillipus
  * License: EPL 1.0
  */

  /*
  * This demonstrates how to open a dialog
  */
  dialog = new Packages.org.tencompetence.tencc.dialogs.CreateUserDialog(window.getShell());
  dialog.open();
}
```

### 1.1.7 DOMs

DOM objects appear as globals in the script namespace. Some DOMs are primitively supported, others are supplied by DOM plug-ins (standard Eclipse plug-ins that contribute an org.eclipse.eclipsemoney.dom extensions). The built-in DOMs available with this release are:

- window - direct access to the Eclipse IWorkbenchWindow object.
- workspace - direct access to the Eclipse IWorkbench object.
- console – direct access to the Monkey Console Viewer

### Writing your own DOM

While the Eclipse APIs are powerful, they can also be a bit overwhelming for typical scripts, thus the Monkey is designed for users to provide their own DOMs (or to use other's DOMs) in addition to the straight Eclipse APIs.

To provide a new DOM, create a new plug-in that extends the org.tencompetence.tencc.monkey.dom extension point. This extension requires two things:

- An implementation of IMonkeyDOMFactory
- The name of the global script variable for the root DOM object

Each script is run in a separate instance of the script engine and the DOM factories are called to create their root objects for each of those instances. The DOM factory returns the root DOM object which can be either a Java object or a org.mozilla.javascript.Scriptable object.

The key to writing a good DOM is to design just the right simple abstractions. It is always tempting to provide everything possible under the sun in one's DOM, but such DOMs turn out to be counter-productive - they are too complex to be easily usable in scripts. We solve the tension between simplicity and features by providing access through our DOMs to the underlying Eclipse API objects; thus when an author finds that our DOM is insufficient, she can reach through to the "real" objects and manipulate the APIs directly.
The TENCompetence plug-in contains a couple illustrative DOM and DOM factory examples.

1.1.8 Links

9.4 Appendix 4 - The TENCompetence QA Component

1.1 Background
The TENCompetence QA Component is a Plug-in for the TENCompetence Client application and can be used to create, edit and deliver multiple choice questions, question pools containing those questions, and questionnaires for performing the questions. These objects can be used for a basic assessment functionality.

The QA Component is one of the first Plug-ins produced for the TENCompetence Client. It plugs into the TENCompetence Datamodel and demonstrates the extensibility of the TENCompetence Client and also provides an examplar for integrating another Datamodel into the TENCompetence data structure.

The author of this Plug-in has experience in the field of e-learning in the area of medical education and administers an ILIAS Learning Management System in the Medical School of Hannover. Since medical students are required to assimilate a great deal of information and take many examinations, they need an opportunity to learn outside of the institution’s physical opening hours. The ILIAS System currently offers an extensive amount of trial modules for those students, including assessment functionality for examination preparation. These questions were created by the medical staff of the Medical School of Hannover and used extensively by the students. The feedback shows that the students consider this functionality to be a very essential e-learning opportunity.

1.1.1 Application Area
The purpose of the QA Component is to provide the user with a quick and easy means to create a pool of questions and for users to answer those questions for self-testing and preparation for examinations. Since questionnaires, question pools and questions can also be obtained from the TENCompetence learning networks, this information can also be shared and of value for other users.

1.1.2 Functionality and Usage

Creation of Objects
The TENCompetence QA Component can be used to create questions, question pools and questionnaires. All those objects are created by wizard dialogs, guiding the author through the individual steps.

Questions serve as the basic element in the QA Component. They contain a question text (e.g. “Who was the Inventor of the light bulb?”) and may contain up to eight answers to this questions which can be either right or wrong (e.g. “Thomas Alva Edison” or “Benjamin Franklin”). The QA Component currently supports two basic question types, these are multiple choice questions with a single possible answer and multiple choice questions with multiple possible answers. The author can also add an internet hyperlink to the question for further reference.
Question pools are the containers of questions. A question exists in one question pool, but can be assigned to several questionnaires. Therefore, if a question pool is deleted, the questions that are in the question pool are deleted as well.

Questionnaires contain links to questions from question pools and not the questions themselves. If a question is edited in a question pool, this will affect each occurrence of this question in every questionnaire. When a questionnaire is deleted, the questions will also stay in the question pools where they belong.

A questionnaire has a specific role in the TENCompetence environment. Whenever a user decides to create a questionnaire, this questionnaire becomes a learning resource. These learning resources will then appear as separate units in the TENCompetence Client, and if questions were previously assigned to them, they can be taken by the user and the QA Component will provide simple and instant feedback on the results in a normal internet browser component.

Displaying the result in a browser component is advantageous since the author of a question is able to add an internet hyperlink to the question they created for further reference. This link is displayed when the user receives feedback on the results. The results browser also links to a forum on the internet, where the users of the questionnaires can come together to discuss and ask more specific questions. This provides the opportunity to bring users together, who may have questions but can’t find an answer in the given internet links.

The Managing of question objects is achieved by means of a Tree View allowing the user to perform the management functions graphically by drag-and-drop and context menus. The Tree View allows question objects to be drag-and-dropped between question pools and questionnaires, moved between questionnaires and also between question pools. It is also possible to sort the questions and change their order in questionnaires and question pools. A context menu enables the user to create new question pools, questionnaires and questions and to delete them. A double click on some components opens an editor window that allows the user to edit the properties of the objects.

Performing of questionnaires
The questionnaires can be displayed in a dialog box where the user can interact with them obtaining, after answering them, their marks and results.

If the user takes a questionnaire, the questions are displayed in a dialog with the possible answers to select from. Once the user has bypassed or answered all the questions, a small browser component is opened to display the results of the assessment to the user, showing the correct or wrongly-answered questions including the correct answers. The component computes the percentage of correctly-answered questions and displays the internet link to a common forum where the user can go to find further help. If the author has linked the questions with a specific internet hyperlink, the user can also choose to select this to get more information about the background of some questions.