Project no. 027087

TENCompetence

Building the European Network for Lifelong Competence Development

Project acronym: Integrated Project TENCompetence

Thematic Priority: 2.4.10

**ID3.24 - Updated design for release 6.0 of the TENCompetence software**

Due date of internal deliverable: 31-10-2009
Actual submission date: 24-12-2009

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

LOGICA

Version 1.0

<table>
<thead>
<tr>
<th>Dissemination Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PU</strong></td>
<td>Public</td>
</tr>
<tr>
<td><strong>PP</strong></td>
<td>Restricted to other programme participants (including the Commission Services)</td>
</tr>
<tr>
<td><strong>RE</strong></td>
<td>Restricted to a group specified by the consortium (including the Commission)</td>
</tr>
<tr>
<td><strong>CO</strong></td>
<td>Confidential, only for members of the consortium (including the Commission Services)</td>
</tr>
</tbody>
</table>
Project Internal Deliverable Report

ID3.24 - Updated design for release 6.0 of the TENCompetence software


Task Task 1: Update the overall TENCompetence architectural design

Date of delivery Contractual: 31-10-2009 Actual: 24-12-2009

Code name ID3.24 Version: 1.0 Draft □ Final ☒

Type of deliverable Report

Security (distribution level) Public

Contributors

Authors (Partner) Hubert Vogten (OUNL), Harrie Martens (OUNL), Stef Heyenrath (LOGICA), Ruud Lemmers (LOGICA), Jules Alberts (OUNL), Anton Finders (OUNL), Leon Schaeps (OUNL)

Contact Person Ruud Lemmers (LOGICA)

WP/Task responsible WP3 / Ruud Lemmers (LOGICA)

EC Project Officer Martin Májek

Abstract (for dissemination) Describes the software architecture of release 6.0 of the PCM, the final release of the project, to developers that need to use, extend or change the PCM server. The views from the 4+1 approach by Kruchten are used to describe the software architecture.

Keywords List Software architecture, design, models, developer guidelines, PCM, services
# Table of contents

1. **Introduction** ........................................................................................................................................... 3  
   1.1. Aim and background ........................................................................................................................... 3  
   1.2. Reading guide ...................................................................................................................................... 3  
   1.3. Acronyms ........................................................................................................................................... 4  

2. **Logical view** ........................................................................................................................................... 5  
   2.1. Introduction ......................................................................................................................................... 5  
   2.2. TENCompetence Portal Architecture ............................................................................................... 6  
      2.2.1. TENCompetence Portlets ........................................................................................................... 8  
      2.2.2. Data model .................................................................................................................................... 11  
   2.3. Authentication ..................................................................................................................................... 16  
   2.4. Authorization ...................................................................................................................................... 16  

3. **Process view** .......................................................................................................................................... 18  
   3.1. Introduction ......................................................................................................................................... 18  
      3.1.1. Sequence diagram for a Get request .......................................................................................... 18  
      3.1.2. Sequence diagram for a Create request .................................................................................. 19  
      3.1.3. Collaboration diagram .............................................................................................................. 20  

4. **Implementation view** ............................................................................................................................ 21  
   4.1. High level overview ............................................................................................................................. 21  
      4.1.1. Liferay Services ............................................................................................................................ 21  
      4.1.2. The project 'org.tencompetence.services-portlet' .................................................................... 22  
      4.1.3. The project 'org.tencompetence.business' .............................................................................. 23  
   4.2. Facades .............................................................................................................................................. 23  
   4.3. Services ............................................................................................................................................. 24  
      4.3.1. Overview .................................................................................................................................... 24  
      4.3.2. Assessment Calculation ............................................................................................................. 24  
      4.3.3. Goal service ............................................................................................................................... 24  
      4.3.4. Learning path service ............................................................................................................... 25  
      4.3.5. Learning plan service ............................................................................................................... 25  
      4.3.6. PDP service ............................................................................................................................... 25  
      4.3.7. Portfolio service ....................................................................................................................... 25  
      4.3.8. Subscribe service ..................................................................................................................... 26  
      4.3.9. UserLevel Calculation .............................................................................................................. 26  
   4.4. Database model ................................................................................................................................... 27  
      4.4.1. CompetenceMap related entities .............................................................................................. 28  
      4.4.2. Assessment related entities ...................................................................................................... 30  
      4.4.3. Other domain entities .............................................................................................................. 31  

5. **Deployment view** .................................................................................................................................... 33  
   5.1. Introduction ......................................................................................................................................... 33  
   5.2. Client layer ......................................................................................................................................... 33  
   5.3. Server layer ....................................................................................................................................... 34  
      Liferay server ...................................................................................................................................... 34  
      TENC Application server .................................................................................................................. 34  
      CopperCore server ............................................................................................................................ 34  

6. **Developer Guidelines** .............................................................................................................................. 35  
   6.1. Introduction ......................................................................................................................................... 35  
   6.2. The Use of a CVS Repository in the TENCompetence Project ....................................................... 35  
      6.2.1. Background ............................................................................................................................... 35  
      6.2.2. CVS and SourceForge .......................................................................................................... 35  
      6.2.3. Access rights and user permissions ........................................................................................ 35  
      6.2.4. Integration with Eclipse ........................................................................................................... 35  
      6.2.5. Repository details .................................................................................................................... 36
6.2.6. Modules and folder structure ................................................................. 36
6.2.7. Other considerations when using folders ............................................. 36
6.2.8. House rules .......................................................................................... 37
6.2.9. Component owners .............................................................................. 37
6.3. Coding conventions and guidelines .......................................................... 37
6.3.1. Naming conventions ........................................................................... 37
6.3.2. Checkstyle ........................................................................................... 39
6.4. Unit testing and JUnit .............................................................................. 39
6.5. Issues and releases managed by Trac ......................................................... 39
6.5.1. Background ........................................................................................ 39
6.5.2. Using Trac .......................................................................................... 39
6.6. Development Environment ...................................................................... 40
6.6.1. Software .............................................................................................. 40
6.6.2. Set up a running Liferay environment in Eclipse ............................... 40
6.6.3. Making an ICEFaces portlet ............................................................... 46
7. Recommended introductory reading .............................................................. 49
8. References ................................................................................................... 50
1. Introduction

1.1. Aim and background

The aim of the document is to describe the software architecture of version 6.0 of the TENCompetence server to developers that need to use, extend or change the TENCompetence server. The different chapters of the document represent different views on the system, each putting their own piece of the puzzle in place. The sections on the ideas and concepts underlying the services will help developers, designers and analysts understand how the system works.

Release 6.0 encompasses the finalization of a complete overhaul of the TENCompetence services infrastructure as a result of some major design decisions and also forms the final release of the infrastructure. The most important part of the overhaul is the close integration with Liferay, a JSR 286 [1] compliant portal environment. The Liferay software provides the integration platform for almost all TENCompetence components. The previous architecture design document ([2]) already mentioned the transition to Liferay, but the current document goes into more detail and explains the benefits of using Liferay.

This document mainly focuses on the TENCompetence services, but also provides portlet development guidelines and discusses the integration issues in some detail.

1.2. Reading guide

Using the concepts from [4] - The 4+1 View Model of Architecture, and adding developer guidelines, the remainder of the document can be divided into five main parts:

1. **Logical view**: gives an overview of all services and explains the main ideas and concepts of the system.
2. **Process view**: shows the system’s most important flows.
3. **Implementation view**: explains the main classes and describes the services in detail.
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 Trac [9] (an issue tracking tool), unit testing, use of CVS (a system for version control of source code), coding conventions and portlet development guidelines.

The use case view from [4] is not described in this document, because it is already covered by WP2 in deliverable D2.4 ([7]).

Chapter 7 – “Recommended introductory reading” Falls outside these five main parts. It provides a few links to documents that are useful when getting started on work related to the TENCompetence project.
# 1.3. Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ajax</td>
<td>Asynchronous JavaScript and XML</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>CAS</td>
<td>Central Authentication Service</td>
</tr>
<tr>
<td>CRUD</td>
<td>Create, Read, Update, Delete</td>
</tr>
<tr>
<td>CSV</td>
<td>Comma Separated Value</td>
</tr>
<tr>
<td>CVS</td>
<td>Concurrent Versions System</td>
</tr>
<tr>
<td>DAO</td>
<td>Data Access Object</td>
</tr>
<tr>
<td>EJB</td>
<td>Enterprise Java Bean</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transport Protocol</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>PCM</td>
<td>Personal Competence Manager</td>
</tr>
<tr>
<td>PDP</td>
<td>Personal Development Planner</td>
</tr>
<tr>
<td>RCP</td>
<td>Rich Client Platform</td>
</tr>
<tr>
<td>SLeD</td>
<td>Service oriented Learning Design</td>
</tr>
<tr>
<td>SSO</td>
<td>Single Sign-on</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>WP</td>
<td>Work Package, one of the groups working in the TENCompetence project</td>
</tr>
<tr>
<td>WP2</td>
<td>Work Package 2 - Requirements and analysis of the integrated system</td>
</tr>
<tr>
<td>WP3</td>
<td>Work Package 3 - Technical design and implementation of the integrated system</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
</tbody>
</table>
2. Logical view

2.1. Introduction

This chapter describes the 6.0 release of the TENCompetence infrastructure.

In discussions leading to the redesign of the TENCompetence infrastructure the following issues with the previous releases were identified:

- An important part of TENCompetence deals with social aspects of users and their communities. Although some social components were provided in previous releases of TENCompetence, such as fora, ratings and chat, in general a need for more and better community tools was identified. Moreover, development of this kind of standard social components are out of the scope of the project.

- The previous versions of TENCompetence provided different software components that were only integrated at the backend through the use of the TENCompetence services. A complete range of applications has been produced varying from desktop tools to web applications. Although this was fine from a pure technical point of view, it provided an unsatisfying user experience. The need for an integrative environment for all these applications was identified. Ideally all applications provided through this integrative environment should share a common look and feel.

The raised issues were addressed by selecting a standard of the shelf portal solution. A portal allows the integration of different components, called portlets, on portal pages. Depending on the situation, this configuration may be done by the individual users or by an administrator.

The Liferay portal was selected to serve as portal container for TENCompetence because Liferay provides some additional features that are very interesting. First of all Liferay provides a number of social portlets out of the box. Furthermore Liferay allows the organization of users to form communities. Finally Liferay is shipped with a complete content management system out of the box.

The decision for using Liferay as the TENCompetence portal had implications for the infrastructure as a whole. For one the applications had to be adapted to support the portlet standard. Two types of adaptation were identified, depending on the level of integration: a loose integration implies that portlets were mere wrappers for existing applications allowing them to be run on the portal. This is the preferred way of integrating any existing application. The second type is much more integrated and reuses the Liferay user management, authentication and authorization. The latter is the preferred choice for new portlets such as part of the desktop applications that still have to be ported to the web. It is clear that this form of integration also has implications for the server architecture and therefore a redesign of the server as it was at the end of 2008 was needed.

Furthermore some issues with the existing server infrastructure were identified:

- An explicit portfolio was missing from the services. Although most information was available in the system, the information was scattered and a client had to combine the portfolio data from different calls. It was concluded that the portfolio deserved a much more central role in the server design.
Some parts of the implementation of the domain model were simplified especially in the area of the competence model. Although this was a deliberate choice, the feeling was that this model was too limited. The full implementation of the domain model was perceived as critical. The domain model was even extended to allow the modelling of competence matrices.

The old service API only allowed access to the personal data of the system. This provided a personal view on the data, but also made it very hard to make non-personalized calls.

After looking closely at the impact of using Liferay as a portal integrative environment in combination with the major impact of the change requests, a service reimplementation was easier than to refactor the existing services. The existing services would be used for the existing applications while any newly developed applications/portlets would make use of the new infrastructure.

At the start of developing release 6.0, a consultancy/training session was organized with an experienced Liferay consultant. He reviewed the architecture of version 5.0. The outcome of this review was that the proposed architecture was feasible and sound in principle. However he pointed out two major concerns:

- Liferay does not offer an active support for the ZK framework and probably will not do so on the short term. Liferay does actively support an equivalent Ajax UI framework called ICEfaces ([13]) and vice versa, ICEfaces actively support Liferay. The big advantage of this construct is support is guaranteed when issues with Liferay and ICEfaces occur, either by Liferay or by ICEfaces;
- Liferay has experience in scaling the portal by load balancing the servers. Support can be provided when using the standard Liferay approach for building services. This could not be guaranteed with the approach chosen in release 5.0, although similar technologies were used.

These two issues led to two major modifications in the architecture:

- The use of ICEfaces as Ajax framework;
- The use of Service Builder for the definition of the model entities and the related services. See section Liferay Service Builder for a brief description of the tool.

This document describes the release of these TENCompetence services.

### 2.2. TENCompetence Portal Architecture

*Figure 1* shows the overall architecture of the TENCompetence infrastructure. The infrastructure can be divided into a number of separate layers. The user interface layer is made up out of portlets complying with the JSR 286 portlet 2.0 specification. These can be deployed and configured on the Liferay portal server. The diagram of *Figure 1* focuses on the way TENCompetence portlets can access the underlying TENCompetence services. The TENCompetence services either access the persistence services provided by Liferay out of the box, or they access the TENCompetence services through a business component.

The TENCompetence business component deals with the business logic of the TENCompetence system. Two major components are implementing this business logic. First, the façade components add validation, journaling and authorization to the underlying entity persistence services. The façade is mainly targeted towards services that directly operate on a
single entity and hides all implementation details involved for the users of the façade. Sometimes the façade itself is not enough, especially in cases of more complex entities. Therefore, secondly, the business component provides services that enhance an entity. A good example of such a service is the LearningPath service. The LearningPath service provides access to the LearningPath entity. A LearningPath [10] defines a set of learning activities that can be ordered and structured in such a manner so they facilitate the user to achieve a specified set competences. The learning path entity is a container for the raw XML data of the learning path. The LearningPath service enhances this entity by providing the marshalling and un-marchalling of the raw XML data into an object model that allows easy manipulation and interpretation of learning paths. It is essential that all portlets using the TENCompetence entities should always use the business component to access them.

The Liferay Persistence Service component depicted in Figure 1 is created via the Liferay Service Builder. On the basis of an XML file defining the entity types, the service creates a database schema, model entity and basic CRUD and search services for manipulation of these entities. The Liferay Persistence Service envelopes services that come out of the box from the Liferay Portal as well as the services provided by TENCompetence. The TENCompetence persistence should never be accessed directly by a portlet, but rather through the façade described in the previous section. However, this cannot be enforced because Service Builder automatically provides portlet access to all services.
2.2.1. TENCompetence Portlets

The 6.0 release of the TENCompetence infrastructure is tightly integrated in the Liferay portal. Not only the user management, authentication and authorization are used from Liferay but also the ICEfaces portlet bridge and the Service Builder are now integral part of the system. However, changes are not only limited to technologies used. The use of portlets also allows a more flexible manner to configure and integrate the TENCompetence application. Rather than merely providing one or more applications, the portlets now provide blocks of functionality that can be configured and combined in several manners. This way it is possible for the administrator of the portal to configure the TENCompetence portlets in a manner that fits the requirements best. For example the administrator of the portal can decide who should have access to particular functionality. Furthermore, the TENCompetence portlets can be combined with other portlets that are available to the portal. Figure 2 shows the portlets that are provided by the release 6.0 of the TENCompetence infrastructure.
Figure 2 Overview of TENCompetence portlets

The portlets are organized based on their main functionality. From the diagram it becomes clear that some portlets are directly dependent on each other, while others are more or less independent. However, the TENCompetence domain model describes how all portlets are related to each other conceptually. Next, a short description per portlet will be given in order to provide a general overview of the available functionality.

<table>
<thead>
<tr>
<th>Portlet</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal Orientation</td>
<td>Provides a user with an overview of the competence development opportunities offered by the learning network. After entering preferences about some competences, the portlet can suggest competence profiles that could be interesting for the user.</td>
</tr>
<tr>
<td>Test</td>
<td>A simple portlet that allows a user to take tests in order to self assess aspects that are relevant for determining a goal. These tests are created using the QTI editor portlet. The outcomes of the test portlet are informative and not directed at a particular competence.</td>
</tr>
<tr>
<td>Learning Path Editor</td>
<td>This portlet provides an editor for creating and maintaining learning paths compliant to the TENCompetence Learning Path specification. The learning paths are used as input to create plans in the personal competence planner.</td>
</tr>
<tr>
<td>Portlet</td>
<td>Short description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Model Editor</td>
<td>This editor creates and manipulates the competence model entities. It includes the definition of competences, competence levels, competence profiles and competence profile levels. These entities are input for all tools that are dealing with competence development.</td>
</tr>
<tr>
<td>QTI Editor</td>
<td>Creates QTI assessments for both the Test portlet and the Assessment portlet.</td>
</tr>
<tr>
<td>Goal Selector</td>
<td>This portlet lets a user choose a goal the user wants to achieve. The selected goal will be input for the Activity Navigator and Assessment portlets. A user can toggle between goals.</td>
</tr>
<tr>
<td>Progress</td>
<td>The Progress portlet (in the past called the “Portfolio portlet”) provides different configurable overviews on user data. User data include progress information about competences, competence profiles, learning activities and evidence. The portlet has two main modes. A “journal” mode, where progress can be monitored in detail and a “current” mode, only presenting the current state of a user. The portlet is very flexible and can serve many purposes.</td>
</tr>
<tr>
<td>Activity Navigator</td>
<td>The Activity Navigator is the portlet that supports a user in actually achieving the selected goal, it thus forms an important part in personal development planning. It consists of two major functionalities. The first is targeted at determining a suitable learning plan for the user based on the available learning paths and the identified competence gaps. The second is targeted at following the learning plan by directing the user to the appropriate learning opportunities.</td>
</tr>
<tr>
<td>Assessment</td>
<td>The assessment portlet helps a user to perform a self assessment for the competences required to achieve the selected goal. After performing this self assessment a competence gap can be determined.</td>
</tr>
<tr>
<td>Evidence</td>
<td>The evidence portlet provides the user with an opportunity to add evidence to the portfolio that provides proof for the mastery of a competence.</td>
</tr>
<tr>
<td>LD Player</td>
<td>The LD Player plays units of learning, that are in the IMS Learning Design format. It is not part of the WP3 work and is therefore not discussed in this document.</td>
</tr>
<tr>
<td>LD Admin</td>
<td>The LD Admin portlet handles administration of units of learning and their runs. It is not part of the WP3 work and is therefore not discussed in this document.</td>
</tr>
</tbody>
</table>
2.2.2. Data model

This section provides a logical view on the data model of the entities used in TENCompetence. The data model can be split into two parts. The first part describes the persistent entities that are generated using Service Builder and that can be accessed through the TENCompetence façade services. The second part is an object model representing a learning path, which is provisioned via the business services. More specifically the learning path model will be discussed in detail as this is a major add-on to the standard persistent entities.

Figure 4 depicts the TENCompetence persistence entities. These entities are generated through Service Builder. TENCompetence clients can access these entities via the façade services and via some of the business services. For the larger part the entities correspond to the entities from the TENCompetence domain model. Most of the presented entities are used to model the TENCompetence components and are not personalized at all. This applies to all entities except three: Goal, Evidence and JournalEntry are used to add user information to the model. The Goal represents the personal motivation of a user to achieve a certain competence profile. The Evidence provides proof underpinning the claim of a user to have mastered a competence. A JournalEntry contains all other user related data. This ranges from logging information about creating, updating, reading and deleting entities to the competences that
have been achieved. The full set of JournalEntries for a user forms the complete journal of the user’s activities in the system. So the journal also provides historic information about the user.

Figure 4 TENCompetence persistence entities

<table>
<thead>
<tr>
<th>Domain Entity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Competence</td>
<td>A Competence entity represents a competence in a CompetenceMap. Competences are scoped with a single CompetenceMap. A Competence entity can have zero or more CompetenceLevels associated with it.</td>
</tr>
<tr>
<td>CompetenceLevel</td>
<td>A CompetenceLevel entity represents a particular level of a Competence. Therefore it is associated with both a Competence and a LevelValue entity. A CompetenceLevel entity can have its own description and title. Furthermore CompetenceLevel entities may contain other CompetenceLevel entities. This way a hierarchy can be formed. Note that this hierarchy is defined for the CompetenceLevel and not for the Competence. Reason for this is that different CompetenceLevels may require different hierarchies.</td>
</tr>
<tr>
<td>LevelValue</td>
<td>A LevelValue entity defines a specific level at which competences can be attained or required. Similar, the LevelValue can be used to define the level for a CompetenceProfileLevel. Each level has its own description and integer ordinal value. The set of LevelValues as specified in the CompetenceMap defines the range of valid levels in the map. Competences and CompetenceProfiles both have their own range of LevelValues defined in the context of a CompetenceMap.</td>
</tr>
<tr>
<td>CompetenceMap</td>
<td>A CompetenceMap entity is the context for all entities related to the competence model: Competence, CompetenceProfile and LevelValue. All these entities are scoped within a</td>
</tr>
<tr>
<td>Domain Entity</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CompetenceMap</td>
<td>A CompetenceMap is associated with a Liferay community.</td>
</tr>
<tr>
<td>Community</td>
<td>A Community is a Liferay construct and is defined as a grouping of users by interest or skill set. For example, a “Pet Lovers” community would consist of users who have an interest in their pets, while a “Tech Support” community would consist of users who have the skills to provide technical support to an organization. A user can belong to any number of communities. Communities have their own pages associated. A Community may be associated with one CompetenceMap.</td>
</tr>
<tr>
<td>CompetenceProfile</td>
<td>A CompetenceProfile represents a job or function and is scoped within the context of a CompetenceMap. A CompetenceProfile can have zero or more CompetenceProfileLevels associated with it.</td>
</tr>
<tr>
<td>CompetenceProfileLevel</td>
<td>A CompetenceProfileLevel defines a CompetenceProfile at a particular level. The level itself is expressed by the LevelValue entity. A CompetenceProfileLevel has a set of CompetenceLevel entities associated with it defining the set of competences and their levels that are required to master the profile.</td>
</tr>
<tr>
<td>Goal</td>
<td>The Goal entity defines targets that are defined or selected by users. Goals have their own descriptions and have a targeted CompetenceProfileLevel. When a user has attained all CompetenceLevels of the CompetenceProfileLevel, the Goals is considered to be achieved.</td>
</tr>
<tr>
<td>Evidence</td>
<td>An Evidence entity contains the proof provided by a user for attaining a competence level. It is therefore associated with a competence level.</td>
</tr>
<tr>
<td>LearningPath</td>
<td>A LearningPath entity captures a learning path. A learning path is expressed through a formalized XML document and its content is stored in the XML data field. Because the LearningPath entity does not interpret this XML, it and can be perceived as a neutral container. A specific data model is provided by the business component to marshal and un-marshal this XML.</td>
</tr>
<tr>
<td>JournalEntry</td>
<td>A JournalEntry captures a single event occurring in the system. Entries are created per user and the set of all entries of a user forms the journal of that user. JournalEntries are never replaced or updated. Instead new entries are added as new events occur, leaving the older entries untouched. The current state of a user is therefore represented by the collection of latest JournalEntries of specific types. All remaining entries represent the history of this user with regard to the system. Therefore it is possible to review the history of a person, which can be filtered on event types and entity types. A JournalEntry has a type that denotes the kind of event that occurred and a value attribute containing the payload of the event. The type determines how that payload should be interpreted. The JournalEntry provides a very powerful and flexible and extensible way of persisting personalized data. The JournalEntry has therefore also a relation with all other entities in the model, which has been left out of the diagram for readability.</td>
</tr>
<tr>
<td>Domain Entity</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AssessmentTest</td>
<td>An IMS QTI compliant test that combines a number of AssessmentItems and determines how the outcomes will be processed. Typical processing includes the mapping of the outcomes to an EQF level. The level can be used later to determine the proficiency level of a user for a competence.</td>
</tr>
<tr>
<td>AssessmentItem</td>
<td>An assessment item is an IMS QTI compliant test item. Items are combined into a single test via the AssessmentTest.</td>
</tr>
</tbody>
</table>

The business component introduces some new data model objects that are not directly persisted. Instead persistence is achieved by marshalling and un-mashalling these objects from the persistence entities. The learning path is the most complex case and will be discussed in more detail. Figure 5 shows the model of the learning path that will be created after marshalling the XML stored in the LearningPath entity. The model is returned as a result of a call to the LearningPath service in the business component.

![Learning Path entities](image)

**Figure 5 Learning Path entities**
<table>
<thead>
<tr>
<th>Domain Entity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LearningPath</td>
<td>The representation of a learning path. The LearningPath has one or more CompetenceLevels associated with it. They are considered to be mastered after successful completion of the LearningPath. The LearningPath can contain three types of Cluster: Sequences, Selections, and Parallel clusters.</td>
</tr>
<tr>
<td>CompetenceLevel</td>
<td>A CompetenceLevel entity represents a particular level of a Competence. Therefore it is associated with both a Competence and a LevelValue entity. A CompetenceLevel entity can have its own description and title. Furthermore CompetenceLevel entities may contain other CompetenceLevel entities. This way a hierarchy can be formed. Note that this hierarchy is defined for the CompetenceLevel and not for the Competence. Reason for this is that different CompetenceLevels may require different hierarchies.</td>
</tr>
<tr>
<td>LearningPlan</td>
<td>A LearningPlan is a generated personal LearningPath. It contains one Selection, which in turn contains one or more LearningPaths that will help the user to achieve the associated Goal. Typically the Activity Navigator is responsible to generate such a LearningPlan for a user. A LearningPlan is always dependent on the Goal. The Goal provides the targets to achieve.</td>
</tr>
<tr>
<td>Goal</td>
<td>The Goal entity defines targets that are defined or selected by users. Goals have their own descriptions and have a targeted CompetenceProfileLevel. When a user has attained all CompetenceLevels of the CompetenceProfileLevel, the Goals is considered to be achieved.</td>
</tr>
<tr>
<td>Sequence</td>
<td>A Sequence is a specialization of a Cluster, which is considered to be completed when all contained LPElements have been completed. The order in which these LPElements can be completed is fixed by the model.</td>
</tr>
<tr>
<td>Selection</td>
<td>A Selection is a specialization of a Cluster, which is considered to be completed when a specified number of contained LPElements have been completed. The order in which these LPElements must be completed is free.</td>
</tr>
<tr>
<td>Parallel</td>
<td>A Parallel is a specialization of a Cluster, which is considered to be completed when a specified number of contained LPElements have been completed. The order in which these LPElements must be completed is free but furthermore these LPElements can be executed in parallel.</td>
</tr>
<tr>
<td>Cluster</td>
<td>A Cluster is an abstract container of LPElements. A Cluster itself is also an LPElement implying that Clusters can be nested.</td>
</tr>
<tr>
<td>LPElement</td>
<td>An LPElement represents an activity or a set of activities that can be performed by user. LPElement can be contained in a Cluster.</td>
</tr>
<tr>
<td>LearningAction</td>
<td>A LearningAction represents an activity a user can perform. Rather that containing the actual content, a LearningAction contains the necessary metadata to guide the user to the content.</td>
</tr>
</tbody>
</table>
2.3. Authentication
Accessing services and entities from the TENCompetence Model require users to identify themselves before being allowed to use the services or to create/read/update and delete entities.

Within the TENCompetence infrastructure, user authentication is handled by Liferay. This means that Liferay will serve as main entry-point for:
- new user registration
- user login (authentication)
- user management (modifying user settings by a user or administrator).

Additionally to offer SSO for the different applications, it is possible to use CAS for authentication. Liferay authentication will then be redirected to the CAS server which will take over the validation of users. When this scenario is used, an additional component needs to be deployed which will handle the user synchronization between Liferay users and CAS users.

2.4. Authorization
Authorization will be handled by Liferay. Liferay provides authorization on the basis of permissions. Permissions are free definable actions and can be defined for a scope. Such a scope is either the enterprise (effectively system wide), a community or an individual. Therefore the current active scope of a user has to be passed via the context. A community acts as the context for the competence model. This implies that a competence map is scoped within a community. The placement of the competence editor portlet on a community page will establish this relationship. Competence maps that are placed on different communities are completely distinct and independent. This way multiple competence maps can be hosted on a single Liferay server. Users can be member of multiple communities and their personal data in the context of a competence map are stored into a single journal using the journal entries. Figure 6 reflects this situation.

![Figure 6 Scoping of the CompetenceMap](image)

Permissions can be set at two distinct levels. The first level operates on actions defined at the portlet level. These permissions typically determine what a user can and cannot do within a portlet. For example the action CREATE_BLOG_ENTRY would determine whether a user has a button allowing the user to create a new blog entry in a blog. The second type of permission is set for a resource. Also actions allowed for a resource can be defined freely and are not limited to a fixed set. Defining permissions on resources allows much more fine grained permissions. TENCompetence uses these types of permissions on its resources via the Security Service.
There are some distinct advantages to re-using the Liferay authorization services. First of all it is already available and does not need to be developed anymore. Of course some effort is needed for integration. Secondly, the user interface provided by Liferay for managing permissions can be reused without any changes. Finally, the user is confronted with a consistent user experience because setting permission for regular Liferay portlets works exactly the same as setting permissions for TENCompetence portlets.
3. Process view

3.1. Introduction

This chapter highlights the main flows of the actual implementation: processing a get request and processing a create request. Both are illustrated via a sequence diagram, focussing on the different layers of the system by using the layers as its swim lanes. The chapter end with a collaboration diagram for the get request, which better shows the links between the different main components.

3.1.1. Sequence diagram for a Get request

*Figure 7* depicts a get request sequence diagram.

![Sequence diagram for a Get request](image)

1. the Security Service is called to check the rights on the entity.
1.1. the Security Service checks the permission for this object with the Liferay server.
2. the Portlet sends a get event to a Service in order to fetch an entity.
2.1. retrieve entity considering business logic roles.
2.1.1. entity is retrieved.
3.1.2. Sequence diagram for a Create request

Figure 8 shows the steps in handling a create request and the involvement of the different application layers.

**Figure 8 Create request sequence diagram**

1. the Security Service is called to check the rights on the entity.

1.1. the Security Service checks the permission for this object with the Liferay server.

2. the Portlet sends a create event to a Service in order to create a new entity.

2.1. create entity considering business logic roles.

2.1.1. the new object is created.

2.1.1.1. a new create event is sent to the MessageQueue.

2.1.1.1.1. an update notification is sent to all services listening to the MessageQueue. The Portfolio Service receives the notification.

2.1.1.1.1.1. a new portfolio entry is created. This entry can be used to determine who created the new entity.

Note: creating a new portfolio entry does not raise another create event. If it would, the system would get stuck in an endless loop.

2.1.1.2. an update notification is sent to the Security Service.

2.1.1.2.1. the Security Service picks up on the update notification and calls Liferay to create a new permission entry for the newly created object.

Finally the newly created entity is returned via the Service to the Portlet.
3.1.3. Collaboration diagram

![Collaboration diagram](image)

Figure 9 Process service request - collaboration diagram

1. the Portlet sends a fetch request for a particular service.
2. the Portlet checks the rights for performing the request with the Security Service.
3. the permissions are checked with the Liferay portal server.
4. the Portlet makes an API call to this Business facade.
5. the Business facade makes an API call to the Service.
6. the appropriate entities are fetched through a find using a convenience method from the DAO classes. Each entity type has its own DAO.
7. the object is retrieved.
8. an event is raised which is redirected to the MessageQueue.
9,10,11,12,13 the MessageQueue informs all registered Services about any changes in the system. It is up to these Service to react accordingly. We already saw that the Security Service will create an entry in the Liferay permission system after a create event. Similar an Indexer will (re)build an index for certain entities when they are created.
4. Implementation view

4.1. High level overview

Figure 10 shows the high level overview of the code projects that together form the TENCompetence system of WP3. In the next sections those projects are described in more detail. The order in which they are described is bottom up, so the lowest layer / project is described first.

![High level overview diagram](image)

4.1.1. Liferay Services

Liferay offers several services which can be used to access Liferay specific functionality like Users, Groups, Companies and Permissions. The code from these services is located in the portal-services.jar, for more information check the (limited) documentation provided by Liferay ([4]).
4.1.2. The project ‘org.tencompetence.services-portlet’

This project is a Liferay portlet which defines the TENCompetence domain model. This 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.

Notes:
1. The first note 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 competence, one for a competence profile and so on. The same applies for relationships.
2. The second note concerns an internal API for creating/reading/updating and deleting these entities. This API is a low-level API which can be used by the higher service-layer. For this the Service Builder tool from Liferay was used to generate the domain model entities, indexes and the services which handle generic functionality to create, read, update and delete an entity or to retrieve multiple entities.

Liferay Service Builder

The TENCompetence domain model is designed using the Service Builder tool from Liferay. This is a tool built by Liferay to automate the creation of interfaces and classes that are used by a given portal or portlet. This includes code for EJBs, Spring, Persistence and Model.

The relations between the domain model entities are not defined in the database, but in the Service Builder generated model and services. The input for the Service Builder is an XML file which defines all model entities, the relations and finders. The syntax from this XML is described in [http://www.liferay.com/dtd/liferay-service-builder_5_2_0.dtd](http://www.liferay.com/dtd/liferay-service-builder_5_2_0.dtd).

Package structure from ‘src’

The packages located in ‘docroot/WEB-INF/src’ contain classes which are only used internally in this project.

<table>
<thead>
<tr>
<th>Packages (src)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.tencompetence.generated.model.impl</td>
<td>The definition of all the DAOs (model entities) which are generated by the Service Builder.</td>
</tr>
<tr>
<td>org.tencompetence.generated.service.impl</td>
<td>All available interfaces and services which are responsible for generic functionality to create, read, update and delete an entity or to retrieve multiple entities.</td>
</tr>
<tr>
<td>org.tencompetence.generated.service.persistence</td>
<td>Generated persistence code and custom code with some custom SQL queries to retrieve Portfolio data from the database.</td>
</tr>
<tr>
<td>org.tencompetence.services.indexers</td>
<td>Classes that process the indexing of model entities. Indexing is done to provide search possibilities in the system.</td>
</tr>
<tr>
<td>org.tencompetence.services.utils</td>
<td>Utility classes that are shared between the model layer and the service layer.</td>
</tr>
</tbody>
</table>
The packages located in ‘docroot/WEB-INF/service’ contain the external API which can be used by other projects.

<table>
<thead>
<tr>
<th>Packages (service)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.tencompetence.generated</td>
<td>The generated classes from the model and services.</td>
</tr>
<tr>
<td>org.tencompetence.services.constants</td>
<td>Classes which define common constants.</td>
</tr>
<tr>
<td>org.tencompetence.services.exceptions</td>
<td>The TENCServiceException class.</td>
</tr>
<tr>
<td>org.tencompetence.services.jaxb</td>
<td>The JAXB generated classes from the LearningPath to provide marshalling and unmarshalling from XML to a LearningPath.</td>
</tr>
<tr>
<td>org.tencompetence.services.xml.qti</td>
<td>XML related classes which handle reading from and writing to XML for creating QTI tests.</td>
</tr>
</tbody>
</table>

4.1.3. The project ‘org.tencompetence.business’

This project defines the high level services and facades. This project is included as a library in all portlets and is the preferred way of accessing the TENCompetence model.

The following table defines the packages.

<table>
<thead>
<tr>
<th>Packages</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.tencompetence.business.constants</td>
<td>Classes which define common constants and enumerations.</td>
</tr>
<tr>
<td>org.tencompetence.business.exceptions</td>
<td>Exception classes for all entities.</td>
</tr>
<tr>
<td>org.tencompetence.business.facade</td>
<td>All facades for generic functionality to create, read, update and delete an entity or to retrieve multiple entities.</td>
</tr>
<tr>
<td>org.tencompetence.business.facade.validators</td>
<td>Validation classes which can be used to validate the properties from an entity when it is added or updated.</td>
</tr>
<tr>
<td>org.tencompetence.business.model.learningpath</td>
<td>The interface and implementation from the LearningPath and elements related to this LearningPath so it can be used more easily in the Activity Navigator.</td>
</tr>
<tr>
<td>org.tencompetence.business.services</td>
<td>The services, in general, contain complex business logic or form the bridge between several facades. For example the PortfolioService combines business logic from several facades.</td>
</tr>
<tr>
<td>org.tencompetence.business.utils</td>
<td>Generic utility classes that are shared between the business project and the portlets.</td>
</tr>
</tbody>
</table>

4.2. Facades

For every entity a service is generated by the Service Builder. Each service offers direct CRUD access to a specific entity. So for example the CompetenceLocalServiceUtil provides functionality to create, read, update and search competences.
In the business project there are several facades. Basically a façade wraps the code from a specific generated entity service and adds some extra functionality. Some examples of extra functionality are:
- Permissions
- Validation
- Logging in Portfolio
- Conversion.

So for example the CompetenceLevelFacade provides the same functionality as CompetenceLocalServiceUtil plus the extras Permissions and Validation. The portlets are strongly urged to use the facades in favor of the generated LocalServiceUtil code.

The following table shows a list from all the facades defined in the business project.

<table>
<thead>
<tr>
<th>Façade</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssessmentItemFacade</td>
<td>Handle CRU + search from QTI assessment items.</td>
</tr>
<tr>
<td>AssessmentTestIdFacade</td>
<td>Handle CRU + search from QTI assessment tests.</td>
</tr>
<tr>
<td>CompetenceFacade</td>
<td>Handle CRU + search from competences.</td>
</tr>
<tr>
<td>CompetenceLevelFacade</td>
<td>Handle RU + search from competence levels.</td>
</tr>
<tr>
<td>CompetenceMapFacade</td>
<td>Handle CRU from competence map.</td>
</tr>
<tr>
<td>CompetenceProfileFacade</td>
<td>Handle CRU + search from competence profiles.</td>
</tr>
<tr>
<td>CompetenceProfileLevelFacade</td>
<td>Handle CRU + search from competence profile levels.</td>
</tr>
<tr>
<td>EvidenceFacade</td>
<td>Handle CRU from user evidence.</td>
</tr>
<tr>
<td>GoalFacade</td>
<td>Handle CRU + search from user goals.</td>
</tr>
<tr>
<td>JournalEntryFacade</td>
<td>Handle CRU + search from portfolio entries and model entities.</td>
</tr>
<tr>
<td>LearningPathFacade</td>
<td>Handle CRU + search from learningpath.</td>
</tr>
<tr>
<td>LevelValueFacade</td>
<td>Handle CRU from level values.</td>
</tr>
</tbody>
</table>

4.3. Services

4.3.1. Overview
In the business project, there are several services defined. Basically a service provides some additional functionality that is not offered by the façade. Examples here are some complex calculations (attained competence level, assessment progress), searching the portfolio and retrieving and searching learning paths and learning plans.

4.3.2. Assessment Calculation
The main goal of this service is to provide information about the assessed competence levels for a certain user's goal.

4.3.3. Goal service
The service provides functionality “select a Goal for a User” and “retrieve a Goal”. In the TENCompetence infrastructure a goal is defined as the ambition of a user to achieve a certain objective. Such an objective can be a competence profile level.
4.3.4. **Learning path service**

The LearningPath service provides access to the LearningPath entity (note: this LearningPath entity is not the same which is provided by the façade but it’s a wrapper around the generated model entity plus some extra functionality).

The main functionality from this service is to provide search functionality to find learning paths for a specified set of competence levels.

A LearningPath ([101]) is created by an experienced author to provide a set of activities in a certain order to facilitate the user to achieve a specific competence profile level. Learning paths and learning actions (i.e. a ‘building block’ in the learning path) can have a more elaborate set of metadata. These metadata are stored within the learning path (XML file) in the database.

4.3.5. **Learning plan service**

The LearningPlan service provides access to the LearningPlan entity. A LearningPlan is a superset from a LearningPath and follows the LearningPath specification to store the selected learning paths for a goal.

4.3.6. **PDP service**

This service is mainly used by the Activity Navigator and provides functionality to retrieve and set the attained competence levels for a user. The other main functionality from this service is to give information about the user’s progress which is related to a certain competence profile level. The information returned is:

1. **Attained**: a percentage which indicates how many competence levels the user has attained for this competence profile,
2. **Planned**: a percentage which indicates how many unattained competence levels are covered by the generated plan and
3. **Not Planned**: a percentage which indicates how many unattained competence levels haven’t been planned yet.

4.3.7. **Portfolio service**

The portfolio service maintains and provides a single point of access to the portfolio data of all users of the TENCompetence infrastructure. The portfolio keeps a record of all artifacts created by users, like evidence added to support proof for attaining a competence at a particular level, but also keeps track of all actions a user has performed whilst using the infrastructure. Typical examples of those actions are creation of a new entity, attaining a level for a competence, completing an activity. Not only can this information be used to show a kind of a log of the actions a user has performed, it also enables the system to show a history of changes to individual entities.

Information is logged to the portfolio service using portfolio entries which store the actions in the system. Each portfolio entry consists of a part containing fixed data and a part containing variable data depending on the type of action being logged. The fixed data is comprised of the date the action is logged, the user performing the action and the id of the entity this portfolio entry is about. The second part of the entry, the variable part, consists out of the type of action and a value parameter. The type is a string whose interpretation is not defined by the portfolio service but by the component responsible for registering the portfolio entry in the first place.
An example of such a component defined type is “target-level” which indicates the action being logged in the portfolio entry is about a user specifying a specific competence level as the study target. The value is also a string whose meaning is defined by the component declaring the action. In the latter example this value parameter would hold the string representation of the competence level identity. To complete the example, the entity Id of this entry would point to the competence for which the user has specified the target level.

From all information stored about a user the portfolio is able to retrieve the current status of this user. This is done by retrieving the latest relevant portfolio entry about the aspect of a user the caller is interested in. So for example when a caller wants to determine the current level at which the user has attained a competence, the portfolio maintained in the course of time a set of entries each indicating the change to a new, probably higher, level of the competence the user had attained at the moment these entries were logged. By looking at the latest portfolio entry of that user, describing details about attaining a competence level, the current status of the user regarding that competence is known. As interpreting the meaning of these entries is depending on both the type and the value of the entries, this interpretation is the responsibility of the caller of the service. The portfolio service merely provides means to collect the raw data from the store of portfolio entries.

Traditionally also profile information of a user like the name, email address and so forth is considered to be part of a portfolio. In the TENCompetence infrastructure however, this information is not maintained by the portfolio itself but by a standard Liferay component, the user profile portlet.

### 4.3.8. Subscribe service

The Subscribe service provides functionality to make it possible for a user to subscribe to a certain entity. This can be a model entity or for example a custom LearningAction entity. Note that the subscribe and unsubscribe events are linked. This means that once the user has subscribed to an entity, it’s not possible to subscribe again. The user first needs to unsubscribe. The subscription information is used by the Activity Navigator to keep track of the current relation and status from the entity and the user.

### 4.3.9. UserLevel Calculation

The main goal of this service is to recalculate the attained competence levels for a user. This method is called when the user attains a competence level. The calculation is done recursively via its child competence levels.

Business rules for calculating the mastery of a competence level:
1) Users have attained competences at a level.
2) If a user has attained a competence at a level, this implies that the user masters this competence at the attained level and all levels below.
3) For a competence a fixed ordered range of competence levels is defined.
4) Competence levels are divided into competences levels containing child competence levels (“folder” competence levels) and competence levels without (“leaf” competence levels).
5) Users can set their attained level (via self assessment or other means) only for leaves. The mastery of folders is always calculated.

6) When setting the level for a folder via a calculation, the level for its children is NEVER changed (no drill down).

7) The portfolio stores the highest attained level for a competence.

Scenario: a user assesses one or more leaves.
When changing the attained level for a leaf, the server needs to calculate the consequences for the folders.

The business logic is implemented like this:

```plaintext
fetch all competence levels in the competence map
loop over all these levels
  if this level is a leaf (no children), skip this level
  if this level is a folder
    attainment = true;
    loop over all children
      if (child is a leaf and user level < required level) then attainment = false
      if (child is a non leaf) check mastery recursively of this child and set attainment accordingly
    end loop
    temporarily store attainment value for this level
  end if
end loop
loop over all competences
  determine highest attained level and store this in the portfolio
end loop
```

4.4. Database model

This section shows the entities used by TENCompetence and explains the meaning of their contents. The entities have been split into three groups: the first group shows the tables related to the Competence Map, the second group shows the tables related to assessments and the third group shows the remaining tables.

All TENCompetence generated model entities have the following common attributes:
- primary key, a identifier which uniquely identifies this entity in the database.
- userId, a reference to the user which created this entity.
- companyId, a reference to the company which was specified when the entity was created.
- groupId, a reference to the community which was specified when the entity was created.

In the domain model, the competence map is the root for several entities. See Figure 11 for an overview of the entities having a direct link to the competence map via the competenceMapId foreign key.

Note: the pictures of the datamodel don’t show foreign key relationships. This is because Liferay, which generates the database tables itself, doesn’t create these. Liferay ensures there are no orphan records.
4.4.1. CompetenceMap related entities

Figure 11 CompetenceMap related entities

**Entity CompetenceMap**

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Title of this entity.</td>
</tr>
<tr>
<td>Description</td>
<td>Long description of this entity.</td>
</tr>
</tbody>
</table>

**Entity LevelValue**

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompetenceMapId</td>
<td>A reference to a CompetenceMap.</td>
</tr>
<tr>
<td>Title</td>
<td>Title of this entity.</td>
</tr>
<tr>
<td>Description</td>
<td>Long description of this entity.</td>
</tr>
<tr>
<td>Value</td>
<td>An integer which defines the level.</td>
</tr>
<tr>
<td>Type</td>
<td>Defines the type of this LevelValue, this can be Competence (0) or CompetenceProfileLevel (1).</td>
</tr>
</tbody>
</table>
Entity CompetenceProfile

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompetenceMapId</td>
<td>A reference to a CompetenceMap.</td>
</tr>
<tr>
<td>Title</td>
<td>Title of this entity.</td>
</tr>
<tr>
<td>Description</td>
<td>Long description of this entity.</td>
</tr>
</tbody>
</table>

Entity CompetenceProfileLevel

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompetenceMapId</td>
<td>A reference to a CompetenceMap.</td>
</tr>
<tr>
<td>Title</td>
<td>Title of this entity.</td>
</tr>
<tr>
<td>Description</td>
<td>Long description of this entity.</td>
</tr>
<tr>
<td>LevelValueId</td>
<td>A reference to the LevelValue to indicate the level of this entity.</td>
</tr>
<tr>
<td>Version</td>
<td>Defines the version of this entity, because multiple versions can exist for a CompetenceProfileLevel.</td>
</tr>
<tr>
<td>VersionTag</td>
<td>Defines the free format tag for this entity.</td>
</tr>
<tr>
<td>ParentId</td>
<td>The identification of the previous version of this CompetenceProfileLevel.</td>
</tr>
<tr>
<td>ChildId</td>
<td>The identification of the next version of this CompetenceProfileLevel.</td>
</tr>
<tr>
<td>CompetenceLevelIds</td>
<td>A CSV list which indicates which CompetenceLevels are related to this CompetenceProfileLevel. Note that the order of these CompetenceLevels is also defined by the order in the CSV list.</td>
</tr>
</tbody>
</table>

Entity Competence

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompetenceMapId</td>
<td>A reference to a CompetenceMap.</td>
</tr>
<tr>
<td>Title</td>
<td>Title of this entity.</td>
</tr>
<tr>
<td>Description</td>
<td>Long description of this entity.</td>
</tr>
<tr>
<td>TestId</td>
<td>A reference to a AssessmentTest.</td>
</tr>
<tr>
<td>Type</td>
<td>Defines the type of this Competence, this can be ‘FUNCTIONAL’, ‘TRANSMETA’ or ‘KNOWLEDGE’.</td>
</tr>
</tbody>
</table>

Entity CompetenceLevel

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompetenceMapId</td>
<td>A reference to a CompetenceMap.</td>
</tr>
<tr>
<td>Title</td>
<td>Title of this entity.</td>
</tr>
<tr>
<td>Description</td>
<td>Long description of this entity.</td>
</tr>
<tr>
<td>LevelValueId</td>
<td>A reference to the LevelValue to indicate the level of this entity.</td>
</tr>
<tr>
<td>Version</td>
<td>Defines the version of this entity.</td>
</tr>
<tr>
<td>VersionTag</td>
<td>A free format name for this version of the entity.</td>
</tr>
<tr>
<td>ParentId</td>
<td>The identification of the previous version of this CompetenceLevel.</td>
</tr>
<tr>
<td>ChildId</td>
<td>The identification of the next version of this CompetenceLevel.</td>
</tr>
<tr>
<td>CompetenceLevelIds</td>
<td>A CSV list to indicate which CompetenceLevels are related to this CompetenceProfileLevel. Note that the order of these CompetenceLevels is also defined by the order in the CSV list.</td>
</tr>
</tbody>
</table>
### Entity Goal

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompetenceMapId</td>
<td>A reference to a CompetenceMap.</td>
</tr>
<tr>
<td>Title</td>
<td>Title of this entity.</td>
</tr>
<tr>
<td>Motivation</td>
<td>Long text describing the motivation for this goal.</td>
</tr>
<tr>
<td>TargetType</td>
<td>Defines the TargetType for this Goal. Currently only ‘org.tencompetence.generated.model.CompetenceProfileLevel’ is supported.</td>
</tr>
<tr>
<td>Target</td>
<td>Defines the unique identifier (of a CompetenceProfileLevel) to which to Goal links.</td>
</tr>
</tbody>
</table>

#### 4.4.2. Assessment related entities

![Assessment related entities](image)

Figure 12 Assessment related entities

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>An identifier for this entity.</td>
</tr>
<tr>
<td>Title</td>
<td>Title of this entity.</td>
</tr>
<tr>
<td>Description</td>
<td>Long description of this entity.</td>
</tr>
<tr>
<td>XmlData</td>
<td>A text field which contains the XML of this test.</td>
</tr>
</tbody>
</table>

```
ID3.24 - Updated design for release 6.0 of the TENCompetence software

**Entity AssessmentItem**

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>A identifier for this entity.</td>
</tr>
<tr>
<td>Title</td>
<td>Title of this entity.</td>
</tr>
<tr>
<td>Description</td>
<td>Long description of this entity.</td>
</tr>
<tr>
<td>XmlData</td>
<td>A text field which contains the XML of this item.</td>
</tr>
</tbody>
</table>

**Table AssessmentTest_AssessmentItem**

This table creates the n:m relation between the AssessmentTest and the AssessmentItems. A test can have multiple items, and an item can be used in several other tests.

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssessmentTestId</td>
<td>The reference to the AssessmentTest.</td>
</tr>
<tr>
<td>AssessmentItemId</td>
<td>The reference to the AssessmentItem.</td>
</tr>
</tbody>
</table>

### 4.4.3. Other domain entities

**Entity Evidence**

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Title of this entity.</td>
</tr>
<tr>
<td>Description</td>
<td>Long description of this entity.</td>
</tr>
<tr>
<td>Date</td>
<td>Defines the date when this Evidence was created or updated.</td>
</tr>
<tr>
<td>TargetType</td>
<td>Defines the TargetType of this Evidence. Currently only ‘org.tencompetence.generated.model.CompetenceLevel’ is supported.</td>
</tr>
<tr>
<td>Target</td>
<td>Defines the unique identifier (of a CompetenceLevel) to which this Evidence contributes.</td>
</tr>
</tbody>
</table>
### ID3.24 - Updated design for release 6.0 of the TENCompetence software

#### Entity LearningPath

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uri</td>
<td>Unique URI for this LearningPath.</td>
</tr>
<tr>
<td>Type</td>
<td>Defines the type of this entity, this can be ‘PATH’ or ‘PLAN’. This is done to make a distinction between a LearningPath and a LearningPlan.</td>
</tr>
<tr>
<td>XmlData</td>
<td>A text field which contains the XML of this LearningPath.</td>
</tr>
<tr>
<td>CoeditorIds</td>
<td>A comma separated value list which indicates which users are allowed to edit this LearningPath.</td>
</tr>
</tbody>
</table>

#### Entity PortfolioEntry

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JournalDate</td>
<td>Defines the date when this entry was created.</td>
</tr>
<tr>
<td>EventType</td>
<td>Defines the type of this event. The table below shows the list of defined event types:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATTAINED_TEST</td>
<td>User has performed an assessment.</td>
</tr>
<tr>
<td>ATTAINED_COMPETENCE_LEVEL</td>
<td>User has attained a CompetenceLevel.</td>
</tr>
<tr>
<td>COMPLETE</td>
<td>User has completed an entity (LearningPath, LearningPlan or Action)</td>
</tr>
<tr>
<td>CREATE</td>
<td>User has created a new entity.</td>
</tr>
<tr>
<td>CREATE_VERSION</td>
<td>User has created a new version of an entity.</td>
</tr>
<tr>
<td>DELETE</td>
<td>User has deleted an entity.</td>
</tr>
<tr>
<td>GOAL_LINKED</td>
<td>User has linked Goal to an CompetenceProfileLevel.</td>
</tr>
<tr>
<td>GOAL_SELECTED</td>
<td>User has selected a Goal.</td>
</tr>
<tr>
<td>LEARNING_PLAN_CURSOR</td>
<td>This indicates where a user is in the execution of a learning plan.</td>
</tr>
<tr>
<td>SUBSCRIBE</td>
<td>User has subscribed to an entity.</td>
</tr>
<tr>
<td>UNSUBSCRIBE</td>
<td>User has unsubscribed from an entity.</td>
</tr>
<tr>
<td>UPDATE</td>
<td>User has updated an entity.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EntityClass</td>
<td>Defines the fully qualified name of the Class which is logged in the PortfolioEntry.</td>
</tr>
<tr>
<td>EntityId</td>
<td>Defines the primary key of the entity which is logged.</td>
</tr>
<tr>
<td>Value</td>
<td>Defines the value which is related to the entity.</td>
</tr>
</tbody>
</table>
5. Deployment view

5.1. Introduction
This chapter describes the deployment of the integrated system. Figure 14 gives an overview of a fully deployed integrated system.

![Deployment overview](image)

Figure 14 Deployment overview

5.2. Client layer
On the client side, there are both web applications and a rich clients. In detail, the client PC needs the following software:

- Rich client: ReCourse (only required for users who will author learning content).
- Web browser: to access all portlets running on the Liferay server or to access a standalone version of LearnWeb.
The rich client can be installed on a computer running either Microsoft Windows, Linux or Mac OS/X. The same goes for the functionality accessed with a web browser.

5.3. Server layer
The second layer consists of server nodes.

Liferay server
The first node is a Liferay Server. This server hosts a Tomcat / Liferay portlet container.

Tomcat runs the Liferay application. Inside this Liferay application, a portlet container is running to serve several portlets. Some portlets are supplied with Liferay, other portlets are created especially by and for the TENCompetence project. These are:
- TENCompetence portlets that connect to a (logical) business layer. The business layer accesses the service-layer (to get access to the domain model) which runs inside Tomcat.
- Portlets that directly access an external CopperCore server (through webservice).
- APIS portlets, whose results can be stored via the TENC portlets, but also use their own database.

A MySQL database management system is hosted for storing all TENCompetence entities. In this deployment, MySQL is installed of the same machine. If desired, MySQL could also be installed on a separate machine.

The appearance of the portlets and Liferay as a whole is set via a tailor made TENCompetence Liferay theme, and for the layout of the portlets in the portal, several additional layout templates were made.

TENC Application server
This server hosts a Tomcat servlet container. Tomcat runs the core components of the old TENCompetence server implementation, because the 1.0 version of LearnWeb uses this for its authentication. When the username/password combinations are kept synchronized between the Liferay and the TENCompetence server database, single sign on for Liferay and LearnWeb can be configured by using CAS ([5]).

A MySQL database management system is hosted for data storage (this could also be installed on another machine). This database is used by Tomcat for storing and retrieving the authorization information. Furthermore the TENCompetenceServer uses the database for its persistence. An Apache webserver running mod_php is used as PHP container for hosting the PHP applications.

CopperCore server
Additionally a CopperCore Server can be installed next to the Liferay and TENCompetence servers. The CopperCore Server hosts a JBoss EJB container on which the SLeD frontend provides access to the TENCompetence version of the CopperCore IMS Learning Design engine ([111]). This CopperCore version is created by the work package “Learning Activities & Units of Learning”.

6. Developer Guidelines

6.1. Introduction
Purpose of this chapter is to explain how the development process is implemented.

6.2. The Use of a CVS Repository in the TENCompetence Project

6.2.1. 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.

6.2.2. CVS and SourceForge
The TENCompetence project uses SourceForge (see [7]) for storage of 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.

6.2.3. Access rights and user permissions
We have allocated key project staff as CVS administrators and assigned 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.

Full documentation for editing the avail file is available at the SourceForge website.

6.2.4. 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.
6.2.5. 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

For an anonymous user:
- Connection type: pserver
- Public User Name: anonymous

For a contributing code developer
- Connection type: extssh
- Public User Name: <user_name_sourceforge>

6.2.6. Modules and folder structure

All code is deposited under the “HEAD” CVS location. WP3 code is placed at the top level under HEAD/portal, 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) followed by the module name, as follows: org.tencompetence.module

Where “modulename” is an appropriate name of the module. For example,
org.tencompetence.all-portlet

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 tencompetence Repository
4. Choose "Use specified module name" which would be wp5/org.tencompetence.myproject

6.2.7. 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, for example. Name the file jdom-1.0.jar, not jdom.jar.

Note that modules cannot be deleted permanently by ordinary users. Only project administrator can delete modules.
6.2.8. **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. 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.
8. Library files should be named according to their version and/or the Eclipse manifest version number set accordingly.

6.2.9. **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 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 tool that requires changes to the central data API of the PDP service, would have to contact Harrie Martens or Hubert Vogten at OUNL. 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.

6.3. **Coding conventions and guidelines**

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

6.3.1. **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;
Fields
Class global Field variables should start with the prefix “f”.
Examples:
• private String fName;
• protected int fCount = 1;

6.3.2. Checkstyle
Checkstyle, as an Eclipse plugin, is used to assist in checking conformance to the style
conditions. It automates the process of checking Java code to spare humans of this tedious
task. This makes it ideal for projects that want to enforce a coding standard.
Each TENCompetence module has a default set of styling conventions defined in an XML file
called “tencompetencecheckstyle.xml”. This file is located in the root of each module.
For the full documentation of CheckStyle see [14].

6.4. Unit testing and JUnit
All code should be covered by unit tests developed using JUnit version 4.x using annotations
for the tests. Code should not be checked if one of the unit tests fails. It is recommended to
have a global unit test incorporating all unit tests of a project for convenience or an equivalent
script. Furthermore it is recommended to run all unit tests after updating a project from CVS.
Unit test should have their own separate source folders in the project. This helps
distinguishing the unit tests from the actual code and makes the building of libraries easier.

6.5. Issues and releases managed by Trac

6.5.1. Background
During final phase of the TENCompetence project, in WP3 issues and releases were managed
by Trac, for the full documentation of Trac see [9]. Trac is an enhanced wiki and issue
tracking system for software development projects. It uses a minimalistic approach to web-
based software project management. Tickets are a portion of work, a reported defect or a
request for change. Every two weeks a new release was published in which a set of tickets
was resolved.

6.5.2. Using Trac
The Trac installation was available at http://trac.tencompetence.org. Because the
TENCompetence project has ended, it’s not guaranteed to remain available.
Developers can request an account by emailing hubert.vogten@ou.nl, but there is also an
account available for visitors:
Username: visitor
Password: tencompetence

New issues were posted in Trac, after this post the issues were assigned to a release. Posting
issues is restricted to registered users (the visitor account doesn’t have this right).
6.6. Development Environment

This section explains how to set up a development environment within the Eclipse IDE, including a setup of Liferay to enable debugging.

6.6.1. Software

The software used during development in TENCompetence was:

- **Eclipse version 3.4.2** as main tooling for the development environment.
- **Java 6 (v1.6.x)** as the SDK, please note that a Java 5 SDK will not work.
- **ICEfaces-1.8.1 plug-in for Eclipse** as the rich internet application framework.
  For the common look and feel of the TENCompetence portlets, the ICEfaces framework was chosen as the rich internet application framework. ICEfaces provides a browser neutral rich user interface experience with extensive use of Ajax while providing developers with a consistent Java API interface. The latter increases efficiency because all regular Java tooling such as integrated development environments and debuggers are available during development.
- **Liferay 5.2.3, running on Tomcat** as the portal.

6.6.2. Set up a running Liferay environment in Eclipse

Set up Liferay and its Plugins SDK

- Download the Bundled with Tomcat 6.0 package of Liferay Portal Standard Edition from [http://www.liferay.com/web/guest/downloads/portal](http://www.liferay.com/web/guest/downloads/portal) and unzip it, for instance to /usr/local/liferay on Linux or C:\bin\liferay on Windows. The remainder of this guide uses {liferay} as placeholder for the folder to which you unzipped this package.
- Download Liferay plugins SDK from [http://www.liferay.com/web/guest/downloads/additional](http://www.liferay.com/web/guest/downloads/additional) and unzip it, for instance to ~/liferay-plugins-sdk-5.2.3 on Linux or C:\liferay-plugins-sdk-5.2.3 on Windows. The remainder of this section uses {pluginsSDK} as placeholder for the folder to which you unzipped this package.
- Create a portal-ext.properties file in the {liferay} folder and give it the following content:
  ```
  jdbc.default.driverClassName=com.mysql.jdbc.Driver
  jdbc.default.url=jdbc:mysql://localhost/lportal?useUnicode=true&characterEncoding=UTF-8&useFastDateParsing=false
  jdbc.default.username=root
  jdbc.default.password=admin
  com.liferay.portal.servlet.filters.gzip.GZipFilter=false
  com.liferay.portal.servlet.filters.header.HeaderFilter=true
  axis.servlet.hosts.allowed=127.0.0.1
  axis.servlet.https.required=false
  ```
- Create a build.(your account name).properties file in the {pluginsSDK} folder and give it the following content: app.server.dir={liferay}/tomcat-6.0.18
Set up Eclipse - Part 1

- Download and install the JEE edition of Eclipse 3.4.2 (via Eclipse Ganymede: http://www.eclipse.org/ganymede/)
- In Eclipse check out the sources from the SourceForge CVS repository. Do this by checking out the following modules from the /portal folder:
  1. org.tencompetence.services-portlet,
  2. org.tencompetence.business,
  3. org.tencompetence.all-portlet (all front-end work will be done in this module),
  4. org.tencompetence.colo-portlet (this one is optional. The portlet can upload profiles and competences from the Dutch Colo initiative. This data is in Dutch.),
  5. org.tencompetence.default-layouttpl and
  6. org.tencompetence.default-theme.

Note: to check out the projects to a custom folder, you need to switch off the option *Use default workspace location* (e.g. don't click Finish in the first screen, but use the Next button until you reach the screen with the *Use default workspace location* checkbox).

- Create a new Tomcat 6 server in Eclipse (File => New => Other => Server, etc.) and use {liferay}/tomcat-6.0.18 as location for the Tomcat installation.

Note: if you encounter timeouts when starting Tomcat, increase the servers timeouts in Eclipse: doubleclick on the server concerned in the Servers tab, go to the *Timeouts* section and increase the values for Start and Stop.

- Update the VM arguments of the Tomcat server (Open launch configuration => Arguments) to:

```
-Dcatalina.base="{liferay}/tomcat-6.0.18"
-Dcatalina.home="{liferay}/tomcat-6.0.18"
-Dwtp.deploy="{liferay}/tomcat-6.0.18/webapps"
-Djava.endorsed.dirs="{liferay}/tomcat-6.0.18/endorsed"
-Xms128m -Xmx1024m -XX:MaxPermSize=256m
-Dfile.encoding=UTF-8
-Duser.timezone=GMT+2
```

Note: the update to the timezone is not required, but it prevents warnings about "deployments in the future".

- In the section *Server Locations* (see Figure 15) select the option *Use Tomcat installation* (takes control of Tomcat installation) and set the *Deploy path* to {liferay}/tomcat-6.0.18/webapps.

- In the section *Publishing* select the option *Never publish automatically*. This is needed because Liferay actions are required when changes are deployed.
Set up Eclipse - Part 2 (to run Flex portlets)

To run the Learning Path editor (or to run other portlets written in Flex) the following additional changes are required:

1. Download the ADOBE FLEX SDK 3.4 via
2. Unzip the SDK to a folder of your own choice.
3. Adapt the build.(your account name).properties file, which is located in the root folder of your Liferay installation. Add a line FLEX_HOME=/dev/flex_sdk_3.4 (replace "/dev/flex_sdk_3.4" with the path to which you unzipped the SDK).
4. Like the Java projects, check out the Flex project
   org.tencompetence.learningpatheditor-portlet.io {pluginsSDK}/portlets.
5. Build the Learning Path Editor by running its build.xml file.
6. Refresh the project (using F5) to copy the .jar's for the business and services projects to the Learning Path Editor project.
Note: for internationalization, the SDK requires a folder for each locale used. Not only within your project, but also in the SDK installation folder. As a result, adding locale nl_NL requires creation of the subfolder /flex_sdk_3.4/frameworks/locale/nl_NL. It doesn't need to contain anything (and the Dutch texts will be part of your project, they don't reside in the subfolder of the SDK), but an empty nl_NL folder has to exist.

Set up Eclipse - Part 3 (Additional Plugins and Configuration)

Add checkstyle 5.x to eclipse.

Use the following steps to enable checkstyle 5.x in Eclipse:

- Add this URL as update site http://eclipse-cs.sourceforge.net/update
- Install the Eclipse Checkstyle Plug-in 5.0.1.200908032119.
- When this plugin is successfully installed, restart Eclipse.
- Now go to configure checkstyle with Window --> Preferences --> Checkstyle
- In the Global Check Configurations add a new Check Configurations
- Use the settings as shown in Figure 16.

![Figure 16 Check Configuration Properties](image)

- Now make sure that the org.tencompetence.all-portlet project uses the correct checkstyle configuration by going to the properties from the project, see Figure 17.
Figure 17 Checkstyle project settings

- Now everything is setup for checkstyle.

Use the correct java code formatter in the project

Check if the TENC code formatter is selected, see Figure 18.
Figure 18 TENC formatter

If not present, press the **import** button and select the xml file: "TENC_CodeFormatterSettings.xml"

**MySQL**

- For Windows machines, download and install MySQL 5.x and its GUI tools from [http://www.mysql.com](http://www.mysql.com).
  For Linux machines, run `sudo apt-get install mysql-server`
- Create a database called `lportal`. Liferay's `portal-ext.properties` file contains a user name and password for MySQL. In the example these were set to the standard values `root` (for user name) and `admin` (for password), you can update them to other values (which is advised from a security point of view).
Development - resolving initial errors

When you open the projects in Eclipse, there are a lot of errors. This is because they depend on code that needs to be generated. To generate the code: right click the build.xml file (located in the root from this project) and choose Run As => Ant Build. This will build the services-project. When a project is built, you need to do a refresh (e.g. via the F5 key) on the project to clear the errors.

The build of the projects needs to be in a specific order:

1. org.tencompetence.services-portlet
2. org.tencompetence.business
3. org.tencompetence.all-portlet

If the Liferay portlets are not correctly deployed when starting Liferay (this is visible by the fact that 'Add Application' shows strange names for one or more TENCompetence portlets), this can be remedied by recompiling the sources while Liferay is running. Liferay will then automatically perform a hot redeploy.

The jar file containing the TENCompetence API is placed in folder {pluginsSDK}/portlets/org.tencompetence.services-portlet/docroot/WEB-INF/lib.

Running

By now you should have a ready environment. Starting the Tomcat server in Eclipse will start Liferay and it will automatically open the opening screen in your default browser.

6.6.3. Making an ICEFaces portlet

This section describes how to create a new portlet called fubar. The description assumes Eclipse, Liferay and the Liferay plugins SDK have been downloaded and installed. If you don't have this yet, read Set up a running Liferay environment in Eclipse.

- Create folder /org.tencompetence.all-portlet/docroot/views/fubar.
- Copy /org.tencompetence.all-portlet/docroot/views/skeleton.jspx to the new folder and rename it to fubar.jspx.
- Make another copy of skeleton.jspx and rename it to addFubar.jspx.
- Create a package /org.tencompetence.all-portlet/docroot/WEB-INF/src/org/tencompetence/portlet/fubar/beans.
- In it create two Java classes: FubarEdit and FubarView.
Create an entry like this in the `/docroot/WEB-INF/faces-config.xml` file:

```xml
<managed-bean>
  <managed-bean-name>fubarEdit</managed-bean-name>
  <managed-bean-class>
    org.tencompetence.portlet.fubar.beans.FubarEdit
  </managed-bean-class>
  <managed-bean-scope>request</managed-bean-scope>
</managed-bean>
```

For navigation add to `faces-config.xml`:

```xml
<navigation-rule>
  <from-view-id>/views/fubar/fubar.jspx</from-view-id>
  <navigation-case>
    <from-outcome>ADD</from-outcome>
    <to-view-id>/views/fubar/addFubar.jspx</to-view-id>
  </navigation-case>
</navigation-rule>
```

Add `<portlet id="Fubar" />` to `/docroot/WEB-INF/liferay-display.xml`

In `/docroot/WEB-INF/liferay-portlet.xml` add:

```xml
<portlet>
  <portlet-name>Fubar</portlet-name>
  <instanceable>true</instanceable>
  <render-weight>1</render-weight>
  <ajaxable>false</ajaxable>
</portlet>
```

In `/docroot/WEB-INF/portlet.xml` add:

```xml
<portlet>
  <portlet-name>Fubar</portlet-name>
  <display-name>Fubar</display-name>
  <portlet-class>com.icesoft.faces.webapp.http.portlet.MainPortlet</portlet-class>
  <init-param>
    <name>com.icesoft.faces.VIEW</name>
    <value>/views/fubar/fubar.iface</value>
  </init-param>
  <supports>
    <mime-type>text/html</mime-type>
    <portlet-mode>view</portlet-mode>
  </supports>
  <portlet-info>
    <title>Fubar</title>
    <short-title>Fubar</short-title>
    <keywords>Fubar</keywords>
  </portlet-info>
  <security-role-ref>
    <role-name>administrator</role-name>
  </security-role-ref>
  <security-role-ref>
    <role-name>guest</role-name>
  </security-role-ref>
  <security-role-ref>
    <role-name>power-user</role-name>
  </security-role-ref>
</portlet>
```
After these steps, the new portlet is complete. Run the build.xml script using Ant and then start Tomcat. In Liferay, when you click Add Application in the "Welcome" menu, the category TENCompetence shows a new portlet Fubar. After adding it, Fubar is shown as a small portlet displaying the text "Skeleton view for TENCompetence development with ICEfaces. Use this ...".

**Internationalization**

- Messages must be in internationalized format. To enable internationalization include in every jspx: `<f:loadBundle basename="org.tencompetence.portlet.util.resources.Language" var="bundle" />`.
- To output a message use `<ice:outputText value="#{bundle["label.description"]}"` /`>.
- And in `/org.tencompetence.all-portlet/docroot/WEB-INF/src/org/tencompetence/portlet/resource/Language.properties` (and `Language_XX.properties`) set the messages like this: `label.title=Title`.
- When editing a properties file, always first do a CVS update, edit quickly and do a CVS commit immediately.
7. Recommended introductory reading

This document assumes readers are familiar with the TENCompetence project and its concepts. People who are completely new to TENCompetence are recommended to study the following besides the current document:

1. The domain model in ([2])
   This document defines the latest version of the logical domain model which underlies all project development work. It shows the model in a class diagram, it has extensive descriptions (including examples) of the classes and it introduces a set of six use cases. The domain model document strongly influenced the WP3 work.

2. Plan for the last year of the project ([11])
   This explains the plan for the last year of the project and from this, one can see what was delivered by the different work packages at the end of the project.

3. TENCompetence project website ([15])
   This website contains a lot of information about the activities of the project and about the background of the project.
8. References


[7] Mendez, C., Arjona, M., Perez, M. (2009), Updated Use Case models based on ID2.11 and Advice on Configuration Management based on D2.17, link not known yet because it is not available in dspace yet.


   Note: at a certain, currently undetermined, point in time the TENCompetence foundation will take over this URL. By then, the content might change.

Links checked on 24-12-2009.