TEN Competence

Building The European Network for Lifelong Competence Development

Building the European Network
For Lifelong Competence Development

TENCompetence IST-2005-027087

Project Deliverable Report

Milestone M5.8 KRSM first cycle prototype evaluation plan

Work Package	Work Package 5
Task	Task 5.3
Date of delivery	**Contractual**: 01-1-2007 **Actual**: 22-1-2007
Code name	**Version**: 1.0 Draft ☒ Final ☐
Type of deliverable	Report
Security (distribution level)	Public
Contributors	Alexander Grigorov (SU), Krassen Stefanov (SU), Marco Luccini (GILABS)
Authors (Partner)	Alexander Grigorov (SU)
Contact Person	Krassen Stefanov (SU)
WP/Task responsible	Marco Luccini (GILABS)
EC Project Officer	Hans-Juergen Westhoff
Abstract (for dissemination)	This report provides a detailed plan for experimentation and evaluation of the KRSM prototype. After the execution of the planned experimentation and evaluation activities, an Evaluation report will be delivered at month 16. This plan will be constantly updated and expanded in line with the planned software development activities related to the KRSM component of the TENCompetence infrastructure.
Keywords List	Evaluation, testing, experimentation, validation, software quality, quality assurance, usability.

TENCompetence Project Coordination at: Open University of the Netherlands
Valkenburgerweg 177, 6419 AT Heerlen, The Netherlands
Tel: +31 45 5762624 – Fax: +31 45 5762800

Revision History

Date	Version	Description	Author
29-09-2006	0.1	First draft version of the report	Alexander Grigorov
03-01-2007	0.5	Complete version with some specific data still missing	Alexander Grigorov, Krassen Stevanov
22-01-2006	0.9	Almost final version	Alexander Grigorov, Krassen Stevanov, Marco Luccini
25-01-2006	1.0	Incorporation of feedback	Alexander Grigorov, Krassen Stevanov, Marco Luccini

Table of Contents

1 Introduction

This document presents the experimentation and evaluation plan of the knowledge resource sharing and management components. This plan will first guide the gap analysis to be performed on the first cycle KRSM releases. After that, the evaluation of the second cycle release will be performed between month 24 and month 28. The evaluation plan will be updated and upgraded according to the emerging needs that may occur during the project period lifespan.

The outcome of these evaluations will be used for the improvement of the system and will be given as input to task 2 of WP5.

2 Methodology

2.1 Objectives and scope

The evaluation of a research project always raises many questions and challenges.

A first set of questions is related to the objective of the evaluation. Is the evaluation conducted to guarantee that the resources have been properly utilized for what they were intended, or is the objective of the evaluation to provide the participants an assessment and some feedback that will help them to better pilot the project and, in particular, maximize the generation of value through this project?

A second set of questions is related to the scope of the evaluation. Are we interested in assessing the process of advancement of the project or in evaluating the quality of the results that are generated by this project? Are we interested in evaluating the technical system (the demonstrator) that is being designed, or in the approach that this system is expected to validate?

An additional set of questions has to do with carrying out the evaluation: What amount of resources should be dedicated to the evaluation of the project? How can we evaluate the effort, and, especially, decide how the evaluation resources are to be allocated? How should we direct the effort (prioritization)? How do we deal with all the risks associated with the evaluation and, in particular, the resistance of people and organizations to participate in an activity that consumes their time, and may threaten their position?

The final set of questions is related to the analysis of the evaluation results and the use of the evaluation. How do we get the most out of this evaluation, identify the most significant results and learn from them?

Answering all these questions is difficult, and is well beyond the scope of this document.

Indeed, if the main focus of a research project should be the maximization of the effectiveness of the evaluation effort in the perspective of the value of the generated knowledge (value for the end user; novelty of the solution; capability to exploit this knowledge), a project very rarely provides the time to evaluate all the potential impacts on the society of the knowledge that has been created.

2.2 Evaluation Criteria

The KRSM system should comply with the following software scope qualitative criteria:

- Relevance: How relevant is the software for the further development of the domain?
- Significance: How important is the problem addressed by the software for the domain? Does the software have a community of users?
- Originality: Are the problems and approaches new? Is this a novel combination of existing techniques?

The software quality of KRSM components should be evaluated using the ISO 9126 [1] quality attributes and guidelines as described in 2.3.

The software coding quality should meet also a number of quality assurance criteria (code readable, code commented, code structured, etc.) as formulated in the TENCompetence Handbook.

The evaluation should also consider:

- the quantity and complexity of the system, identifying clearly new developments and re-usability of existing components;
- the impact of the system (what is the added value, is it downloaded from the CVS from other users, what is their opinion, etc.);
- the extent the KRSM components meet the functional requirements of the system.

2.3 Which quality attributes will be tested

ISO 9126 [1] gives guidelines and describes the quality attributes that could be used for the evaluation of a software product. The ISO 9126 model defines six product characteristics (see Figure 1):

- functionality;
- reliability;
- usability;
- efficiency;
- maintainability;
- portability.

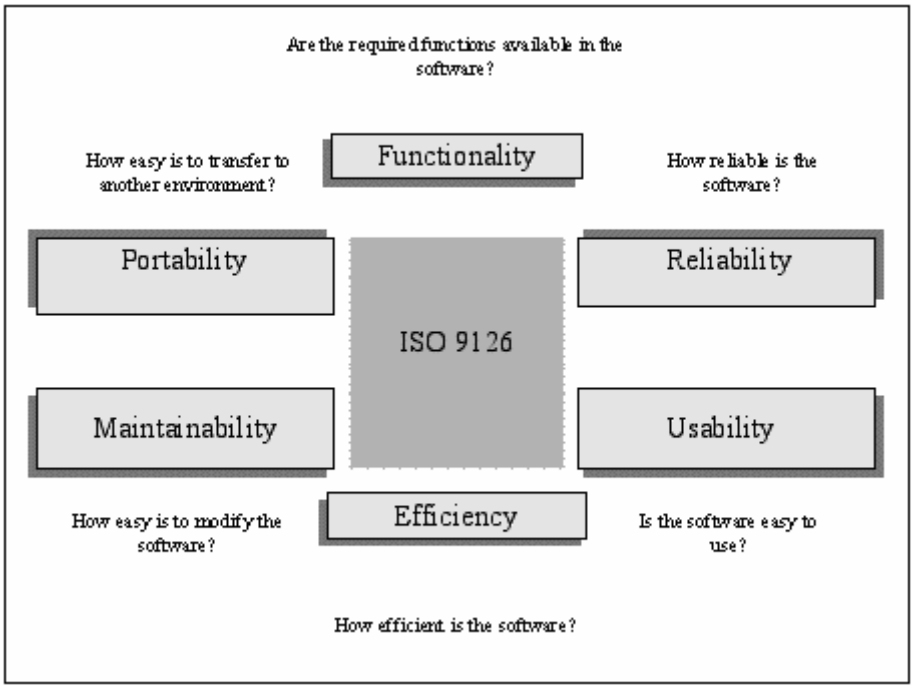

Figure 1: ISO 9126:1991

These six characteristics (attributes) are further subdivided into a number of sub-characteristics. Table 1 presents the quality attributes and their description.

Quality attributes	Description: the capability of the software product to…
Functionality	**provide functions which meet stated and implied needs when the software is used under specified conditions.**
Suitability	provide an appropriate set of functions for specified tasks and user objectives.

Accuracy	provide the correct or agreed results or effects with the needed degree of precision.
Interoperability	interact with one or more specified systems.
Security	protect information and data so that unauthorised persons or systems cannot read or modify them and authorised persons or systems are not denied access to them.
Functionality compliance	adhere to standards, conventions or regulations in laws and similar prescriptions relating to functionality.
Reliability	**maintain a specified level of performance when used under specified conditions.**
Maturity	avoid failure as a result of faults in the software.
Fault tolerance	maintain a specified level of performance in cases of software faults or of infringement of its specified interface.
Recoverability	re-establish a specified level of performance and recover the data directly affected in the case of a failure.
Reliability compliance	adhere to standards, conventions or regulations relating to reliability.
Usability	**be understood, learned, used and attractive to the user, when used under specified conditions.**
Understandability	enable the user to understand whether the software is suitable, and how it can be used for particular tasks and conditions of use.
Learnability	enable the user to learn its application.
Operability	enable the user to operate and control it.
Attractiveness	be attractive to the user.
Usability compliance	adhere to standards, conventions, style guides or regulations relating to usability.
Efficiency	**provide appropriate performance, relative to the amount of resources used, under stated conditions.**
Time behaviour	provide appropriate response and processing times and throughput rates when performing its function, under stated conditions.
Resource utilisation	use appropriate amounts and types of resources when the software performs its function under stated conditions.
Efficiency compliance	adhere to standards or conventions relating to efficiency.
Maintainability	**be modified. Modifications may include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications.**

Analysability	be diagnosed for deficiencies or causes of failures in the software, or for the parts to be modified to be identified.
Changeability	enable a specified modification to be implemented.
Stability	avoid unexpected effects from modifications of the software.
Testability	enable modified software to be validated.
Maintainability compliance	adhere to standards or conventions relating to maintainability.
Portability	**be transferred from one environment to another.**
Adaptability	be adapted for different specified environments without applying actions or means other than those provided for this purpose for the software considered.
Installability	be installed in a specified environment.
Co-existence	co-exist with other independent software in a common environment sharing common resources.
Replaceability	be used in place of another specified software product for the same purpose in the same environment.
Portability compliance	adhere to standards or conventions relating to portability.

Table 1: ISO 9126 Quality Attributes.

Particular attention should be paid to the evaluation of the usability of KRSM components.

ISO 9241-11 Guidance on Usability [2] further extends the definition of **Software Usability**. According to ISO 9241-11 (1998), usability is the "extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use."
Important aspects of this definition include:

- **"specified users"**: It is important to note that when systems are being designed for usability, the first step should be identifying target user population. Usability is not an absolute term but, rather, a relative one. A system can only be usable relative to the user population it serves.
- **"specified goals"**: The functionality designed into a usable system will be relevant to its users. It is often the case that unnecessary or inappropriate functionality is incorporated into a system. This functionality can "clutter" the interface and make relevant functionality more difficult to access. On the other hand, a usable

system presents its users with routes to achieve their goals in a clear fashion.

- **"effectiveness"**: Effectiveness is the accuracy and completeness with which users achieve specified goals. For example, a software (SW) is effective if the users can complete tasks making a minimal amount of errors.
- **"efficiency"**: Efficiency is the resources expended in relation to the accuracy and completeness with which users achieve specified goals. For example, a SW is efficient if the users are able to achieve their goals quickly (saving time) or cheaply (saving money).
- **"satisfaction"**: Satisfaction is the freedom from discomfort and positive attitudes towards the use of the product. For example, a user is satisfied, if (s)he is able to achieve her/his own goals with a positive frame of mind. The user may also prefer one particular system to other systems.
- **"context of use"**: The context of use constitutes the broader framework in which a product is operated. It concerns the system's particular users, their tasks and the system's broader environment of use.

2.4 Types of Testing

Testing can be done on a number of different levels:

- unit/module test;
- integration test;
- functional test;
- system test;
- acceptance test.

Unit test

Unit testing searches for defects in, and verifies the functioning of, software (e.g. modules, programs, objects, classes, etc.) that are separately testable.
Unit tests are typically done by programmers and not by testers, as it requires detailed knowledge of the internal program design and code. The ideal situation is that another developer then the developer from the software runs the unit test.

The purpose of unit testing is to verify that each individual component functions according to the technical specifications. Unit testing includes several subjects, namely:

- Completeness of each unit
- Correct processing by unit
- Relation controls within the unit
- Correct execution of each unit
- Integrity of the database
- Applying standards in case of error handling, logging and such
- Menu structure, short keys
- Screen navigation
- Field controls, value ranges, maximal precision, field length
- Mutation or non-mutation of the proper fields at the right time
- Error handling
- Association with next/previous unit
- Performance of the (components of the) unit

Integration test

The purpose of integration testing is to verify if the interaction between the components of the system works correctly. There are several subjects to consider within integration testing:

- Interfaces between units in an application
- Complete processing chain
- Relation controls within the system including several modules and/or in combination with the database

Functional test

Functional testing is based on analysis of the specification of the

- Security testing;
- etc.

Acceptance test

Formal testing with respect to user needs, requirements, and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to accept the system.

3 Evaluation of the KRSM system

3.1 Objectives and scope

Objectives

The objectives of this evaluation plan are:
1. To check the quantity and complexity of the system, identifying clearly new developments and re-usability of existing components;
2. To check the quality of the software system (see 2.3);
3. To check the impact of the system (what is the added value, is it downloaded from the CVS from other users, what is their opinion …)
4. To check to what extent the KRSM components meet the functional requirements of the system.

Scope

Testing will be done on functional and system test level. The lower levels (unit testing and integration testing) are performed by the development team of the KRSM system during the development process, so these tests do not fall within the scope. This document focuses only on the system and functional level tests of the KRSM system.

First, after the first prototype is developed, quality and quantity tests will be performed. Later on, after finishing the first release prototype, impact and functional testing will be performed.

3.2 Test approach

The approach and methodology for testing are following the recommendations given in TENCompetence D4.1 "Pilot evaluation plan". The structure of the testing and evaluation plan is following the structure proposed in Appendix 4 of D4.1.

Test design process steps

The complete test design process consists of the following sub-steps:

1. Collect the software documentation (preferably a functional design, if not available gather information from other sources (publications, working documents, forum info, own knowledge).
2. Make a test plan.
3. Define the scope (e.g. only the calculating functions and not the personal administration will be tested and which test methods used, e.g. decision table).
4. Determine clusters.
5. Determine test conditions
6. Determine test case(s)

Test implementation process steps

These are the practical steps in order to perform the testing in real settings. The following sub-steps are identified:

7a. Organize the test environment.

7b. Make documentation of the configuration of the test environment (to make the test repeatable).

8. Execute the tests and record the results of each test. This also includes retesting fixed defects.
9. Make a report of the test results (to report on the number of failed and passed tests).

3.3 Evaluation Schedule

Tasks	Start Date	End Date
Quantity & Complexity	5.02.2007	25.02.2007
Quality of the software	15.02.2007	10.03.2007
Test Preparation		
Preparing Test Plan	01.09.2007	15.01.2007
Design of Test Clusters, Conditions and Cases	01.11.2007	15.01.2007

Preparing Test Data	15.01.2007	20.02.2007
Test Implementation		
Test Environment Setup	20.02.2007	05.03.2007
Test Execution	05.03.2007	20.03.2007
Evaluation Report	15.02.2007	31.03.2007

3.4 Roles and Responsibilities

We can identify the following roles within TENCompetence regarding testing:

- *The test manager*: the test manager is responsible for the testing within a project.
- *The test team leader*: the daily management of a test project can be handed over to a test team leader. The final responsibility for the test remains, however, with the test manager. The role of the test team leader is similar to that of foreman in the test team, ensuring that everything is organized so that the team is able to carry out its work effectively.
- *The test analyst/executor*: creates the test design and carries out the test cases. He will build up the test set based on the product risks and requirements pertaining to the information system. The test analyst should have knowledge about and experience in the domain area.

The evaluation of the proof-of-concept version of KRSM that will focus only on the quantity and quality of the software system will be performed by experts - TENCompetence partners that are not involved in WP5.

Test team involved with the evaluation of the KRSM system is the following*:

No	Name	Partner	Role
1			Test Manager
2			Team Leader
3			Test Analyst/Executor
4			Test Analyst/Executor
5			Test Analyst/Executor

*The test team will be determined by 31.01.2007.

3.5 Test Environment

Test environment includes definition, organization and documentation of the hardware and software environment used for the experiments. This includes for example:

- Server-side services (federated search, rating, integration of external services like Learn eXact, TASTE, Flickr and YouTube, Authorisation and Authentication, Publishing, Downloading, etc.)
  - Installation
  - Hardware
  - Systems
  - Capabilities
- Client-side services (local search, share, create, store, GUI, etc.)
  - Installation (KRSM Client)
  - Hardware (processor: at least 2 MHz, Intel or AMD based; memory – 512 MB; HD – at least 100 MB free, connection to Internet)
  - Software (OS: Linux, MS Windows; JVM installed)

The test environment should be documented, so the experiments could be repeated.

The test environment should include a number of computers connected via KRSM P2P network.

3.6 Evaluation of the Quantity and Complexity of the System

The following questions should be answered in order to assess the quantity and complexity of the KRSM system:

- What existing knowledge resource sharing and management tools and components have been used (for example: LionShare P2P network, ARIADNE and DSpace repositories, Flickr, YouTube, etc.)?
- Are these tools and components open source and what licenses do they have?
- What is their functional description and technical characteristics?
- What is the KRSM system architecture?
- What are the new developments?
- How many source code lines have been written?

- How many and what Java modules, classes and interfaces have been developed?
- What APIs have been specified and documented?
- How are the KRSM components integrated?

At the end the total Output points for the KRSM tool (according to the TENCompetence Handbook) should be calculated.

3.7 Evaluation of the Quality of the Software

For the evaluation of the quality of the software we have defined a general questionnaire given in Appendix 6.1. It uses a 5 point Likert scale and is based on the guidelines and the quality attributes as described in ISO 9126 [1] (see 2.3).

To assess the software coding quality we have defined a set of specific questions that need to be answered during the evaluation process. These questions are given in the questionnaire in Appendix 6.2 and are following the quality assurance criteria as described in the TENCompetence Handbook. The criteria include:

- Code readable
- Code commented
- Code structured
- Code efficient
- Testing
- Deployment
- API documentation available
- Source code publicly available with archive facility
- Licensed and download available.

The results of the questionnaire should be described and analysed in the Quality report, and used for the further improvement of the KRSM tool.

3.8 Design of Test Clusters, Conditions and Cases

This kind of testing is intended to show how KRSM tool is fulfilling the identified user requirements, according to the scenarios and use cases described.

For the design of test clusters, conditions and cases we are using the methodology described in TENCompetence D4.1 “Pilot evaluation plan” and briefly presented below.

Each test level can be divided into four stages: Preparation, Analysis, Navigation (optional, used for automatic testing) and Execution.

At each stage of test development, the question is: what should be tested and how do we know if the test is reliable enough? Each test can be divided into logical blocks, or 'Test Clusters' in order to achieve a greater degree of reliability. These Test Clusters give the tests a logical structure. Next, a number of Test Conditions are defined within each Test Cluster. These are elaborated into concrete Test Lines (also called Test Cases), which form the transition into the testing itself. The clarity, which this testing structure produces, improves the ease of maintenance and reuse of the test products.

The TestFrame (the LogicaCMG method) Excel sheets can be used as templates for the description of Test Clusters, Test Conditions and Test Cases. These templates are given in Appendix 6.3.

The Test cases are derived from the WP5 scenarios and use cases. They are grouped in 4 Test Clusters: 'Authoring', 'Sharing', 'Storing' and 'Access and Usage' that correspond to the main components of the KRSM system.

The following Test clusters, conditions and cases are designed for the evaluation of the KRSM system:

Cluster 001: Authoring

Test Condition 001C1: Create a knowledge resource

Test Case 001C1T1: Create a knowledge resource with a resource editor / authoring tool

Test Case 001C1T2: Create a knowledge resource via resource format selection (ordinary flow)

Test Case 001C1T3: Create a knowledge resource via resource format selection (exception 1)

Test Case 001C1T4: Create a knowledge resource via resource format selection (exception 2)

Test Case 001C1T5: Create a knowledge resource via resource format selection (exception 3)

Test Case 001C1T6: Create a knowledge resource via resource format selection (exception 4)

Test Case 001C1T7: Create a knowledge resource via resource format selection (exception 5)

Test Condition 001C2: Delete of a knowledge resource

Test Case 001C2T1: Delete of a knowledge resource

Cluster 002: Sharing

Test Condition 002C1: Share a knowledge resource

Test Condition 002C1T1: Share a knowledge resource by setting the permission parameters
Test Condition 002C1T2: Share a knowledge resource by setting the access parameters (ordinary flow)
Test Condition 002C1T3: Share a knowledge resource by setting the access parameters (exception 1)

Cluster 003: Storing

Test Condition 003C1: Store a knowledge resource

Test Case 003C1T1: Store a knowledge resource (ordinary flow)
Test Case 003C1T2: Store a knowledge resource (exception 1)
Test Case 003C1T3: Store a knowledge resource (exception 2)
Test Case 003C1T4: Store a knowledge resource (exception 3)

Test Condition 003C2: Add a new repository

Test Case 003C2T1: Add a new repository

Cluster 004: Access and Usage

Test Condition 004C1: Browse resources

Test Case 004C1T1: Browse resources (ordinary flow)
Test Case 004C1T2: Browse resources (exception 1)

Test Condition 004C2: Search resources

Test Case 004C2T1: Search resources
Test Case 004C2T2: Search resources (exception 1)
Test Case 004C2T3: Search resources (exception 2)

Test Condition 004C3: Access (retrieve) a resource

Test Case 004C3T1: Access (retrieve) a resource

Test Condition 004C4: Rate a quality of a resource

Test Case 004C4T1: Rate a quality of a resource

The descriptions of the test cases are given in Appendix 6.4. The test cases are designed for the functional test of the KRSM system but can be used also for making test scenarios for the other planned experiments.

3.9 Test Execution

The execution of the tests will be performed by a number of test analysts/executors following the test cases given in Appendix 6.4. During the testing the test executors should carefully record all test results, errors, problems and observations.

The test executors should answer the questions described in 3.6 in order to evaluate the quantity and the complexity of the software. For the evaluation of the quality of the software they should also review the source code and the documentation and fill in the questionnaires given in Appendices 6.1 and 6.2.

After the test completion the test executors should also fill in the KRSM Functionality Questionnaire given in Appendix 6.5 by indicating the implementation state for each functionality (e.g. fully implemented, implemented but needs improvement, partially implemented, not implemented) and give some comments, suggestions or observations.

3.10 Evaluation Report

The final Evaluation Report should include the results of the performed questionnaires, expert reviews and analyses, tests, test summary, and test analysis. The report should also include gap analysis by providing a general statement of the capability of the system as demonstrated by the test, compared with the requirements, stating the system deficiencies and recommending improvements of the system.

4 Next steps

According to the steps described in the previous chapter, the Evaluation plan will be constantly updated.
First we have to determine the test team - expected deadline 31.01.2007.

Moreover, one of the most important activities to be performed in the forthcoming months (in particular, before the end of the first cycle of the project) is the gap analysis of the tests cases in order to prevent possible mismatches due to the cyclic updates of WP5 use cases. Therefore, the comparison and contrast evaluation procedure will be updated according to the changes in the WP5 use cases. Expected deadline - 14.02.2007.

After that we should prepare a detailed test time schedule - who should perform what test and when. Expected deadline - 28.02.2007.
In February and March we have to perform execution of all questionnaires and tests in line with the planned time schedule.

In parallel with that we have to prepare the Evaluation Report - expected deadline 31.03.2007.

After the end of the first cycle we have to perform a usability test with end users - expected deadline 30.09.2007.

5 References

[1] International Standard ISO/IEC 9126. Information technology -- Software product evaluation -- Quality characteristics and guidelines for their use, International Organization for Standardization, International Electrotechnical Commission, Geneva.

[2] International Standard ISO 9241-11 (1998), Ergonomic requirements for office work with visual display terminals (VDTs) - Part 11 : Guidance on usability

6 Appendices

6.1 General Questionnaire on the Quality of the Software

		Strongly disagree				Strongly agree	
Functionality		1	2	3	4	5	Comments
Suitability	The software can perform the required tasks						
Accurateness	The result is as expected						
Interoperability	The system can interact with another system						
Security	The software prevents unauthorized access						
Reliability		1	2	3	4	5	Comments
Maturity	Most of the faults in the software been eliminated over time						
Fault tolerance	The software is capable of handling errors						
Recoverability	The software can resume working and restore lost data after failure						
Usability		1	2	3	4	5	Comments
Understandability	The system enables the user to understand how to use the system easily						
Learnability	The user can learn to use the system easily						
Operability	The user can use the system without much						

	effort						
Attractiveness	The interface looks good and attractive to the user						
Efficiency		1	2	3	4	5	Comments
Time Behaviour	The system responds quickly enough						
Resource Utilization	The system utilizes resources efficiently						
Maintainability		1	2	3	4	5	Comments
Analyzability	Faults can be easily diagnosed						
Changeability	The software can be easily modified						
Stability	The software can continue functioning if changes are made?						
Testability	The software can be easily tested						
Portability		1	2	3	4	5	Comments
Adaptability	The software can be moved to other environments						
Installability	The software can be installed easily						
Conformance	The software complies with portability standards						
Replaceability	The software can easily replace other software						

6.2 Coding Quality Questionnaire of the KRSM System

Question	Yes	No	N/A	Comments
Are all the APIs needed for the integration in WP3 client available?				
Is Object Oriented programming used correctly (e.g. is there clear separation of concerns)?				
Is all the code placed in the TENCompetence CVS repository?				
Is there proper use of try/catch and managing of exceptions?				
Are Java classes properly documented using Javadoc (or Javadoc like style)?				
Are there appropriate inline comments?				
Does the source code follow Java source code conventions (e.g. Eclipse, D3.1 naming conventions for classes, interfaces, methods, variables, etc.) ?				
Are variable names human readable?				
Is source code well structured?				
Is there proper indentation of code?				
Does source code contain any dead code?				

Does source code contain any hard coded values?				
Is code efficient?				
Is JUnit used for unit test framework?				
Can the software be installed and run on a different environment than the one it is used for development??				
Is there API documentation available?				
Is English used properly in code and documentation?				
Is the User Interface modelled after the Eclipse User Interface Guidelines?				

6.3 Templates for Test Clusters, Conditions and Cases

Cluster card

Cluster	**<clustername>**	cluster ID	**XXX**	(3 character abbreviation)
Sheet	**Cluster chart**	date last modified	**<date>**	
Version	**<version>**	modified by	**<name>**	
Date	**<date>**			
Author	**<author>**			
KEY INFORMATION				
Clustername	*<clustername>*			
System	*<Systemname and version>*			
Test type	*<In which testkind will the cluster be used?>*			
ASSIGNMENT				
Risks	*<Which product risks can occur in case this cluster is not executed or when errors occur when in production?>*			
Importance	*< What is the relative importance of this cluster as opposed to other clusters? The importance results from the product risks determined. The importance determines testing priority and planning. Use the MOSCOW standard: 'must test', 'should test', 'could test' and 'won't test'. By determining the cluster importance using the product risk, the test manager avoids clusters being assigned the highest priority by stakeholders. Separate prioritising can be assigned to test conditions at a later stage, with high risk projects. >*			
Quality attribute	*<Which quality attributes are addressed in this cluster?>*			

Base documentation	*<A reference to documentation should be made on which the test design for this cluster is based. Additional information concerning elimination or reduction of product risks should be added in case default documentation does not provide this.>*		
EXECUTION			
Test approach	*<How is the test executed? A test manager's choice depends on test type, quality attribute, available documentation, organization, circumstances etc. Statical tests: auditing and reviewing? Or use of dynamic testing: decision tables, entity life cycle, data flow analysis? Manual or automated test execution?>*		
Test environment	*<Which environment is needed to execute the specified test? Both technical and mantime resources as well as other dependencies should be mentioned. Is a production-like environment needed or is a make-and-break environment enough? Which test data is needed?>*		
RESULT			
Acceptance criteria	*< When does the stakeholder accept the cluster? Make sure the acceptance criteria are stated explicitly. The criteria should be known to all parties involved when the test project has finished. >*		

Test conditions

Cluster	**<clustername>**	cluster ID	**XXX**
Sheet	**test conditions**	date last modified	**<date>**
Version	**<version>**	modified by	**<name>**
Date	**<date>**		
Author	**<author>**		
number of test conditions	**4**		

This document contains the test conditions for cluster <clustername>

Test condition number	Test condition description	Priority	Status
XXXC1	Description test condition 1	High	not started
XXXC2	<description test condition n>	must test	not started
XXXC3		must test	
XXXC4			

Test cases

Cluster	**<clustername>**	**cluster ID**	**XXX**	
Sheet	**Test cases nn**	**date last modified**	**<date>**	
Version	**<version>**	**modified by**	**<name>**	
Date	**<date>**			
Author	**<author>**			
number of test cases	**3**			
Precondition for the test				
test condition	**XXXC1**	**Description test condition 1**		
test case	**XXXC1T1**	**Description of test case 1**		
Precondition				
Actions				
Expected result				
Actual result				
test case	**XXXC1T2**	**Description of test case 2**		
Precondition				
Actions				
Expected result				
Actual result				

6.4 KRSM Test Cases

Cluster	**Authoring**	**cluster ID**	**001**	-
Sheet	**Test cases**	**date last modified**	**01.01.2007**	-
Version	**0.1**	**modified by**	**Sofia University**	-
Date	**01.01.2007**			-
Author	**Sofia University**	-	-	-
number of test cases	**8**	-	-	-
Precondition for the test				
test condition	**001C1**	**Create a knowledge resource**		
test case	**001C1T1**	**Create a knowledge resource with a resource editor / authoring tool**		
Precondition	1. The needed resource doesn't exist 2. A set of resource editors / authoring tools must be available. 3. The User has opened the KRSM UI.			
Actions	1. The User selects the 'Create a Resource' entry in the Menu Bar of the KRSM UI. 2. The User selects the proper resource editor / authoring tool from a list. 3. The resource editor / authoring tool is loaded. 4. The resource editor / authoring tool is opened. 5. The User creates the resource needed.			
Expected result	1. The needed resource has been created. 2. The resource is loaded into the KRSM system. 3. The list of 'Most recently-used' resources is updated.			
Actual result				
test case	**001C1T2**	**Create a knowledge resource via resource format selection (ordinary flow)**		
Precondition	1. The needed resource doesn't exist 2. A set of resource editors / authoring tools must be available.			

	3. The User has opened the KRSM UI.
Actions	1. The User selects the 'Create a Resource' entry in the Menu Bar of the KRSM UI. 2. The User selects the kind of resource (e.g. Text, Image, Video, Audio, Animation, Presentation) from a list. 3. A list of suitable tools (i.e. the resource editors) by which the resource can be produced and a list of resource formats that can be associated to the selected kind of resource are offered to the User. a. According to the kind of resource selected a 'default' / 'most recommended' tool among the ones available is highlighted b. According to the kind of resource selected a 'default' / 'most recommended' format among the ones available is highlighted 4. The User selects the resource editor / authoring tool from a list. 5. The resource editor / authoring tool is loaded. 6. The resource editor / authoring tool is opened. 7. The User creates the resource needed.
Expected result	1. The needed resource has been created. 2. The resource is loaded into the KRSM system. 3. The list of 'Most recently-used' resources is updated.
Actual result	
test case	**001C1T3 Create a knowledge resource via resource format selection (exception 1)**
Precondition	1. The needed resource doesn't exist 2. A set of resource editors / authoring tools must be available. 3. The User has opened the KRSM UI.
Actions	1. The User selects the 'Create a Resource' entry in the Menu Bar of the KRSM UI. 2. The User selects the kind of resource (e.g. Text, Image, Video, Audio, Animation, Presentation) from a list. 3. A list of suitable tools (i.e. the resource editors) by which the resource can be produced and a list of

	resource formats that can be associated to the selected kind of resource are offered to the User. a. According to the kind of resource selected a 'default' / 'most recommended' tool among the ones available is highlighted b. According to the kind of resource selected a 'default' / 'most recommended' format among the ones available is highlighted 4. The User selects the resource format from the list. 5. A list of suitable tools (i.e. the same one as before or a restricted one, depending on the resource selected) by which the resource can be produced is offered to the User. a. According to the format selection a 'default' / 'most recommended' tool among the ones available is highlighted 6. The User selects the resource editor / authoring tool from a list. 7. The resource editor / authoring tool is loaded. 8. The resource editor / authoring tool is opened. 9. The User creates the resource needed.
Expected result	1. The needed resource has been created. 2. The resource is loaded into the KRSM system. 3. The list of 'Most recently-used' resources is updated.
Actual result	
test case	**001C1T4 Create a knowledge resource via resource format selection (exception 2)**
Precondition	1. The needed resource doesn't exist 2. A set of resource editors / authoring tools must be available. 3. No available editing tools that can be associated to the selected kind of resource. 4. The User has opened the KRSM UI.
Actions	1. The User selects the 'Create a Resource' entry in the Menu Bar of the KRSM UI. 2. The User selects the kind of resource (e.g. Text, Image, Video, Audio, Animation, Presentation) from a list. 3. The KRSM returns no results available, that is, no tools available that can be associated to the selected

	kind of resource. 4. The KRSM suggests: a. to select another kind of resource; b. to download and install a proper tool. 5. The User selects another kind of resource from a list. 6. A list of suitable tools (i.e. the resource editors) by which the resource can be produced and a list of resource formats that can be associated to the selected kind of resource are offered to the User. a. According to the kind of resource selected a 'default' / 'most recommended' tool among the ones available is highlighted b. According to the kind of resource selected a 'default' / 'most recommended' format among the ones available is highlighted 7. The User selects the resource editor / authoring tool from a list. 8. The resource editor / authoring tool is loaded. 9. The resource editor / authoring tool is opened. 10. The User creates the resource needed.
Expected result	1. The needed resource has been created. 2. The resource is loaded into the KRSM system. 3. The list of 'Most recently-used' resources is updated.
Actual result	
test case	**001C1T5 Create a knowledge resource via resource format selection (exception 3)**
Precondition	1. The needed resource doesn't exist 2. A set of resource editors / authoring tools must be available. 3. No available editing tools that can be associated to the selected kind of resource. 4. The User has opened the KRSM UI.
Actions	1. The User selects the 'Create a Resource' entry in the Menu Bar of the KRSM UI. 2. The User selects the kind of resource (e.g. Text, Image, Video, Audio, Animation, Presentation) from a list.

	3. The KRSM returns no results available, that is, no tools available that can be associated to the selected kind of resource. 4. The KRSM suggests: a. to select another kind of resource; b. to download and install a proper tool. 5. The User downloads and installs the editing tool. 6. The resource editor / authoring tool is loaded. 7. The resource editor / authoring tool is opened. 8. The User creates the resource needed.
Expected result	1. The editing tool has been downloaded and installed. 2. The list of editing tools locally-available is updated. 3. The needed resource has been created. 4. The resource is loaded into the KRSM system. 5. The list of 'Most recently-used' resources is updated.
Actual result	
test case	**001C1T6 Create a knowledge resource via resource format selection (exception 4)**
Precondition	1. The needed resource doesn't exist 2. A set of resource editors / authoring tools must be available. 3. No available editing tools that can be associated to the selected kind of resource. 4. The User has opened the KRSM UI.
Actions	1. The User selects the 'Create a Resource' entry in the Menu Bar of the KRSM UI. 2. The User selects the kind of resource (e.g. Text, Image, Video, Audio, Animation, Presentation) from a list. 3. The KRSM returns no results available, that is, no tools available that can be associated to the selected kind of resource. 4. The KRSM suggests: a. an alternative resource format (e.g. no .mov editors available, but .avi ones);

	b. to download and install a proper tool. 5. The User selects the alternative resource format. 6. The User selects the resource editor / authoring tool from a list. 7. The resource editor / authoring tool is loaded. 8. The resource editor / authoring tool is opened. 9. The User creates the resource needed.
Expected result	1. The needed resource has been created. 2. The resource is loaded into the KRSM system. 3. The list of 'Most recently-used' resources is updated.
Actual result	
test case	**001C1T7 Create a knowledge resource via resource format selection (exception 5)**
Precondition	1. The needed resource doesn't exist 2. A set of resource editors / authoring tools must be available. 3. No available editing tools that can be associated to the selected kind of resource. 4. The User has opened the KRSM UI.
Actions	1. The User selects the 'Create a Resource' entry in the Menu Bar of the KRSM UI. 2. The User selects the kind of resource (e.g. Text, Image, Video, Audio, Animation, Presentation) from a list. 3. The KRSM returns no results available, that is, no tools available that can be associated to the selected kind of resource. 4. The KRSM suggests: a. an alternative resource format (e.g. no .mov editors available, but .avi ones); b. to download and install a proper tool. 5. The User downloads and installs the editing tool. 6. The resource editor / authoring tool is loaded. 7. The resource editor / authoring tool is opened. 8. The User creates the resource needed.

Expected result	1. The editing tool has been downloaded and installed. 2. The list of editing tools locally-available is updated. 3. The needed resource has been created. 4. The resource is loaded into the KRSM system. 5. The list of 'Most recently-used' resources is updated.	
Actual result		
test condition	**001C2**	**Delete of a knowledge resource**
test case	**001C2T1**	**Delete of a knowledge resource**
Precondition	1. The resource is available. 2. The User has opened the KRSM UI. 3. The User has retrieved the resource.	
Actions	1. The User selects the 'Properties' entry in the Menu Bar of the KRSM UI.. 2. The User flags the resource as 'Deleted'.	
Expected result	1. The resource is no more available. 2. The list of 'Most recently-used' resources is updated.	
Actual result		

Cluster	**Sharing**	**cluster ID**	**002**
Sheet	**Test cases**	**date last modified**	**01.01.2007**
Version	**0.1**	**modified by**	**Sofia University**
Date	**01.01.2007**		
Author	**Sofia University**		
number of test cases	**3**		
Precondition for the test			
test condition	**002C1**	**Share a knowledge resource**	
test case	**002C1T1**	**Share a knowledge resource by setting the permission parameters**	
Precondition	1. The resource is available. 2. The User is allowed to access the resource. 3. The User has the rights to modify the access to the resource. 4. The User has opened the KRSM UI. 5. The User has retrieved the resource		
Actions	1. The User selects the 'Properties' entry in the Menu Bar of the KRSM UI. 2. The User sets the kind of access to the resource, that is, the permissions parameters (e.g. RWX). 3. Optionally, the User can set the kind of user is allowed to access the resource by selecting it from a list		
Expected result	1. The access rights to the resource are updated. 2. The access rights to the resource are stored into the KRSM system. 3. The list of 'Most recently-used' resources is updated.		
Actual result			
test case	**002C1T2**	**Share a knowledge resource by setting the access parameters (ordinary flow)**	
Precondition	1. The resource is available. 2. The User is allowed to access the resource. 3. The User has the rights to modify the access to the resource. 4. The User has opened the KRSM UI.		

	5. The User has retrieved the resource
Actions	1. The User selects the 'Properties' entry in the Menu Bar of the KRSM UI. 2. The User sets the kind of user is allowed to access the resource by selecting it from a list. 3. Optionally, the User can set the kind of access to the resource, that is, the permissions parameters (e.g. RWX)
Expected result	1. The access rights to the resource are updated. 2. The access rights to the resource are stored into the KRSM system. 3. The list of 'Most recently-used' resources is updated.
Actual result	
test case	**002C1T3 Share a knowledge resource by setting the access parameters (exception 1)**
Precondition	1. The resource is available. 2. The User is allowed to access the resource. 3. The User has the rights to modify the access to the resource. 4. The User has opened the KRSM UI. 5. The User has retrieved the resource 6. The entered user has to be registered in the TENCompetence system.
Actions	1. The User selects the 'Properties' entry in the Menu Bar of the KRSM UI. 2. The User sets the kind of user is allowed to access the resource by selecting it from a list. 3. Optionally, the User can set the kind of access to the resource, that is, the permissions parameters
Expected result	1. The access rights to the resource are updated. 2. The access rights to the resource are stored into the KRSM system. 3. The list of 'Most recently-used' resources is updated.
Actual result	

Cluster	**Storing**	**cluster ID**	**003**
Sheet	**Test cases**	**date last modified**	**01.01.2007**
Version	**0.1**	**modified by**	**Sofia University**
Date	**01.01.2007**		
Author	**Sofia University**		
number of test cases	**5**		
Precondition for the test			
test condition	**003C1**	**Store a knowledge resource**	
test case	**003C1T1**	**Store a knowledge resource (ordinary flow)**	
Precondition	1. The resource is available. 2. A set of (local and / or remote) repositories onto which to store the resource is available. 3. The User is allowed to access the set of repositories. 4. The User has opened the KRSM UI. 6. There is enough free space on the repository / ies to store the resource.		
Actions	1. The User selects the 'Store a Resource' entry in the Menu Bar of the KRSM UI. 2. The User selects the resource to be stored and loads it into the KRSM system. 3. The System shows the User a list containing the (local and remote) repositories which she / he is allowed to access. 4. The User selects the target repository / ies where to store the resource 5. The User stores the resource.		
Expected result	1. The needed resource has been stored. 2. The list of 'Most recently-used' resources is updated		
Actual result			
test case	**003C1T2**	**Store a knowledge resource (exception 1)**	
Precondition	1. The resource is available. 2. A set of (local and / or remote) repositories onto which to store the resource is available.		

	3. The User is allowed to access the set of repositories. 4. The User has opened the KRSM UI. 5. There is not enough free space on the repository / ies to store the resource.
Actions	1. The User selects the 'Store a Resource' entry in the Menu Bar of the KRSM UI. 2. The User selects the resource to be stored and loads it into the KRSM system. 3. The System shows the User a list containing the (local and remote) repositories which she / he is allowed to access. 4. The User selects the target repository / ies where to store the resource 5. The KRSM system detects that the resource is too bulky to be stored on the selected repository. But enough space can be freed on the Repository. 6. The needed space on the Repository is *automatically* freed 7. The User stores the resource.
Expected result	1. The needed resource has been stored. 2. The list of 'Most recently-used' resources is updated
Actual result	
test case	**003C1T3 Store a knowledge resource (exception 2)**
Precondition	1. The resource is available. 2. A set of (local and / or remote) repositories onto which to store the resource is available. 3. The User is allowed to access the set of repositories. 4. The User has opened the KRSM UI. 5. There is not enough free space on the repository / ies to store the resource.
Actions	1. The User selects the 'Store a Resource' entry in the Menu Bar of the KRSM UI. 2. The User selects the resource to be stored and loads it into the KRSM system. 3. The System shows the User a list containing the (local and remote) repositories which she / he is allowed to access. 4. The User selects the target repository / ies where to store the resource

	5. The KRSM system detects that the resource is too bulky to be stored on the selected repository. But enough space can be freed on the Repository only if the Repository's MAX_STORAGE_THRESHOLD is increased. 6. The Repository's MAX_STORAGE_THRESHOLD is *automatically* increased up to the needed value 7. The User stores the resource.
Expected result	1. The needed resource has been stored. 2. The list of 'Most recently-used' resources is updated
Actual result	
test case	**003C1T4 Store a knowledge resource (exception 3)**
Precondition	1. The resource is available. 2. A set of (local and / or remote) repositories onto which to store the resource is available. 3. The User is allowed to access the set of repositories. 4. The User has opened the KRSM UI. 5. There is not enough free space on the repository / ies to store the resource.
Actions	1. The User selects the 'Store a Resource' entry in the Menu Bar of the KRSM UI. 2. The User selects the resource to be stored and loads it into the KRSM system. 3. The System shows the User a list containing the (local and remote) repositories which she / he is allowed to access. 4. The User selects the target repository / ies where to store the resource 5. The KRSM system detects that the resource is too bulky to be stored on the selected repository and enough space can not be freed on the Repository.
Expected result	1. The needed resource has not been stored. 2. The KRSM system displays a message to the User that there is no enough space on the repository and suggests splitting the Resource into smaller chunks or changing the storage capacity of the repository.
Actual result	

test condition	**003C2**	**Add a new repository**
test case	**003C2T1**	**Add a new repository**
Precondition	1. The new repository is available. 2. The User has got the Access rights to the repository. 3. The User has got the rights to perform KRSM Administration activities. 4. The User has opened the KRSM UI.	
Actions	1. The User selects the 'Administration' entry in the Menu Bar of the KRSM UI. 2. The User selects the 'Add a repository' entry. 3. The User selects a new repository to add. 4. The repository has been added.	
Expected result	1. The new repository has been mapped into the KRSM system. 2. The new repository has been made available into the KRSM system. 3. The list of repositories has been updated.	
Actual result		

Cluster	**Access and usage**	**cluster ID**	**004**
Sheet	**Test cases**	**date last modified**	**01.01.2007**
Version	**0.1**	**modified by**	**Sofia University**
Date	**01.01.2007**		
Author	**Sofia University**		
number of test cases	**7**		
Precondition for the test			
test condition	**004C1**	**Browse resources**	
test case	**004C1T1**	**Browse resources (ordinary flow)**	
Precondition	1. A set of (local and / or remote) repositories onto which to search the resource is available. 2. The User is allowed to access the set of repositories. 3. The User has opened the KRSM UI.		
Actions	1. The User selects the 'Most recently-used' entry in the Menu Bar of the KRSM UI. 2. The User browses the 'most recently used' resources list to see whether the resource is already loaded (i.e. 'cached') into the KRSM system or not		
Expected result	1. The needed resource has been sought. 2. If the resource is found and selected the list of 'Most recently-used' resources is updated		
Actual result			
test case	**004C1T2**	**Browse resources (exception 1)**	
Precondition	1. A set of (local and / or remote) repositories onto which to search the resource is available. 2. The User is allowed to access the set of repositories. 3. The User has opened the KRSM UI.		
Actions	1. The User selects the 'Open…' entry in the Menu Bar of the KRSM UI. 2. The User browses the resources available in the network of repositories.		

Expected result	1. The needed resource has been sought. 2. If the resource is found and selected the list of 'Most recently-used' resources is updated	
Actual result		
test condition	**004C2**	**Search resources**
test case	**004C2T1**	**Search resources**
Precondition	1. A set of (local and / or remote) repositories onto which to search the resource is available. 2. The User is allowed to access the set of repositories. 3. The User has opened the KRSM UI.	
Actions	1. The User selects the 'Search…' entry in the Menu Bar of the KRSM UI. 2. The User enters the name or part of it (e.g. by using wildcard characters), format extension included, of the resource to be sought into a search field. a. Optionally, the User can set the maximum number of results fetched by the search differently from the 'default' value set and proposed by the system. b. Optionally, the User can set the maximum number of results displayed at once differently from the 'default' value set and proposed by the system. c. Optionally, the User can set some additional search parameters related to resource sharing / access permission, quality rating, activity history, and so on.	
Expected result	1. The needed resource has been sought. 2. The KRSM system returns a list of results. 3. If the Preview is set 'ON' in the KRSM UI Preferences > Search results, the resources are previewed too. 4. The list of results is cached	
Actual result		
test case	**004C2T2**	**Search resources (exception 1)**
Precondition	1. A set of (local and / or remote) repositories onto which to search the resource is available. 2. The User is allowed to access the set of repositories.	

	3. The User has opened the KRSM UI.
Actions	1. The User selects the 'Search…' entry in the Menu Bar of the KRSM UI. 2. The User selects a content type. a. Optionally, the User can set the maximum number of results fetched by the search differently from the 'default' value set and proposed by the system. b. Optionally, the User can set the maximum number of results displayed at once differently from the 'default' value set and proposed by the system. c. Optionally, the User can set some additional search parameters related to resource sharing / access permission, quality rating, activity history, and so on. 3. The User submits the search request. 4. The KRSM opens a basic content editor. 5. The User edits and submits the content pattern.
Expected result	1. The needed resource has been sought. 2. The KRSM system returns a list of results. 3. If the Preview is set 'ON' in the KRSM UI Preferences > Search results, the resources are previewed too. 4. The list of results is cached
Actual result	
test case	**004C2T3** **Search resources (exception 2)**
Precondition	1. A set of (local and / or remote) repositories onto which to search the resource is available. 2. The User is allowed to access the set of repositories. 3. The User has opened the KRSM UI.
Actions	1. The User selects the 'Search…' entry in the Menu Bar of the KRSM UI. 2. The User sets search parameters related to resource sharing / access permission, quality rating, activity history, and so on. a. Optionally, the User can set the maximum number of results fetched by the search differently from the 'default' value set and proposed by the system.

	b. Optionally, the User can set the maximum number of results displayed at once differently from the 'default' value set and proposed by the system. 3. The User submits the search request. 4. The KRSM opens a basic content editor. 5. The User edits and submits the content pattern.	
Expected result	1. The needed resource has been sought. 2. The KRSM system returns a list of results. 3. If the Preview is set 'ON' in the KRSM UI Preferences > Search results, the resources are previewed too. 4. The list of results is cached	
Actual result		
test condition	**004C3**	**Access (retrieve) a resource**
test case	**004C3T1**	**Access (retrieve) a resource**
Precondition	1. The resource is available. 2. The User is allowed to access the resource. 3. The User has opened the KRSM UI.	
Actions	1. The User selects the 'Search…' entry in the Menu Bar of the KRSM UI. 2. The User seeks the resource. 3. The User selects the resource. 4. The User loads the resource (e.g. by double clicking). 5. The resource is retrieved (i.e. cached in the KRSM system).	
Expected result	1. The needed resource has been retrieved. 2. The list of 'Most recently-used' resources is updated	
Actual result		
test condition	**004C4**	**Rate a quality of a resource**

test case	004C4T1 Rate a quality of a resource
Precondition	1. The resource is available. 2. The User is allowed to access the resource. 3. The User has the rights to vote the resource. 4. The User has opened the KRSM UI. 5. The User has retrieved the resource.
Actions	1. The User selects the 'Properties' entry in the Menu Bar of the KRSM UI. 2. The User sets the Quality of the resource by: a. entering a value in a proper field (where range values are provided by the KRSM UI) b. selecting a value in a list (e.g. a Likert scale, whose range end values are provided by the KRSM UI).
Expected result	1. The resource Quality rating is updated. 2. The resource Quality rating is stored into the KRSM system. 3. The list of 'Most recently-used' resources is updated
Actual result	

6.5 KRSM Functionality Questionnaire

Please, fill in the following questionnaire by indicating the implementation state for each functionality (e.g. fully implemented, implemented but needs improvement, partially implemented, not implemented) and give some comments, suggestions or observations.

Functionality	Implementation State	Comments / Observations
Create a Resource		
Search for / retrieve a resource		
Store a resource		
Pack / build knowledge resources		
Delete a resource		
Modify access and permission rights associated to a resource (Sharing)		
Rate a resource		
Preview a resource		
Preview resource's metadata		
Log in / out the KRSM system		
Set the working mode		
Add a repository		
Set the storage capacity		

Perform back-up		
Recommending systems for users' preferences via collaborative filtering like 'Taste'		
Sharing images via 'Flickr'		
Sharing videos via 'YouTube'		
Sharing bookmarks via 'Technorati'		
Creating / sharing text documents and spreadsheets via 'Google Docs & Spreadsheets'		