Project Number: 032518

QVIZ

Query and context based visualization of time-spatial cultural dynamics

Specific Targeted Research Project

Information Society Technologies

Software validation diary and final validation report D7.1.2

Due date of deliverable: 31/01/2008

Actual submission date: 11/04/2008

Start date of project: 01/05/2006 Duration: 24 month

Umeå University Final

Abstract

Number and name	Project Number: 032518 QVIZ, Query and context based visualization of time-spatial cultural dynamics
Work Package	WP7
Task	D 7.1.2
Date of delivery	**Contractual:** 31/01/2008 **Actual:** 11/04/2008
Code name	032518 **Version** draft ☐ final ☑
Nature	Report + On-line diary http://qviz.humlab.umu.se/index.php/TEST_PAGE http://qviz.humlab.umu.se/flyspray/
Distribution Type	Public
Authors (Partner)	Peder Andrén, peder.andren@svar.ra.se Mona Bonta Bergman, mona.bontabergman@humlab.umu.se Johan Lagrelius, johan.lagrelius@humlab.umu.se Linda Lindgren, linda.lindgren@ladok.umu.se Edith Seegel, edith.seegel@ra.ee Bhawana Srivastava, bhawana.srivastava@ladok.umu.se Kristina Teral, kristina.teral@ra.ee Tõnis Türna, tonis.tyrna@ra.ee
Contact Person	Linda Lindgren, linda.lindgren@ladok.umu.se
Abstract	Report on the final functionality testing and software validation for the QVIZ project, co-funded by ICT Research Framework Programme of the European Union.
Keywords List	Validation, Use Case, Bug report

Table of contents

1. Introduction

The main objectives of the final validation report is to validate that the development of the QVIZ software meets the specifications and user functional requirements as outlined in the Description of Work in the Technical Annex.

One of the main outputs of the functional testing besides this report has been to prepare for the usage scenarios and case study for the usability tests that supersede the functionality testing.

This deliverable can be seen as the third part of linked deliverables that validates the systems compliance to the specifications and the priorities set in the preceding deliverables, D4.1.3 System Specification and requirement report and D7.3 Assessment methodology and User scenario test plan. The fourth and final report D 7.4 will contain the case study report and usability test results.

This report describes the methodology and the test results from the functional software evaluation of the two prototypes implemented between M19 to M22 of the project. Prototype 2a was delivered in month 19 and prototype 2b was delivered in month 22.

The validation report also includes a bipartite on-line diary that follows the progress of the testing:

The first part of the diary is a wiki page that gathers all the documentation from the testing. It holds the latest version of all documents and notes from the testing sessions, thereby creating a transparent test organization where the recent findings and comments are readily available for both the testers and the developers of the system. This page can be found on the QVIZ wiki at the following address: http://qviz.humlab.umu.se/index.php/TEST_PAGE

The second part of the on-line diary is a bug reporting system called Flyspray that effectively works as a communication interface between the test organization and the developers. The bugs are reported in detail by the testers and assigned to the developer that needs to fix the bug. The reporting system also works as an effective forum where the users easily can add comments and questions related to the issues described in the bug reports. The bug reporting tool can be found at the following address: http://qviz.humlab.umu.se/flyspray/

The methodology has changed somewhat after the finalization of the D7.3. This document describes the reasons for this divergence; a description of the chosen methodology and why the new method was to be preferred in the QVIZ project as well as a comparison between the different methods are presented. An account for the way the chosen methodology has been translated to QVIZ testing will be given, as well as the results that have been reported to the developers along with the results from the validation. The report will also provide recommendations for future implementation. In addition to the functionality validation, a short summary will be provided on the validation of content.

2. Validation of system functionality

2.1 Objectives of validation

The objective of Work Package 7 is to validate the functionality of the deliverables Prototype 2a & Prototype 2b and the QVIZ Integrated Platform. This work package also verifies that the user requirements from deliverable D 4.1.3 are met, that the system works according to the technical specifications (also from D 4.1.3), and includes the intended features defined in the earlier WP2 and WP4. Validation means to identify problems, find bugs, irregularities and see to user friendliness issues before launching the system to the public and presenting it to the focus groups that are set via the target groups.

Overlooking the importance of validation makes it nearly impossible to identify and mitigate all critical risks that may threaten the value of the final results of this project. Validation is needed to assure that all important quality factors that were decided upon by the consortium are met. The requirements, needs and expectations must be followed up and validated by intensive testing.

The Validation verifies that the complete system serves its purpose and fulfills the requirements of a helpful tool for the various focus groups.

From the Description of Work one can derive that one of the objectives within WP7 is validation of software and specifications. That it is needed to ensure continuous review and assessment of the project results and more importantly to provide feedback to the implementation activities. Evaluations regarding the specifications of the software must be made through validation of the presented prototypes. To do this, there needs to be a test set up that can elaborate the progression, regression and verification in order to validate the system functionalities. The Description of Work also states that all validation steps will be fed into the software cycle.

The functionality objectives that can be derived from the above text have been fulfilled through using the previous documentation in the test formation, validating content and through the methodology chosen.

2.2 Methodology for functionality testing

In order to optimize the validation of the platform a strategic decision was made concerning the test methodology to be used for validating the product. After a careful review of the documentation, the status of the project, pressing time limits and the changes made to the projects final outcome, it was decided to choose a slightly different approach than the initially intended test plan described in D7.3. The test team chose to work towards a strategy more suitable for this project in the existing conditions. The employment of an experienced test coordinator to see the functionality testing through made it clear that the above change of methodology was needed. The strategy chosen is a method called Exploratory Testing and is described in further detail in the next section.

The main reasons for choosing Exploratory Testing are firstly that the method is optimized to find bugs fast and provides a quick turn around of feedback to the developers and management in an interactive process. Secondly that it allows the testers and developers to spend more time testing and developing and less time on preparing documentation. The method has over the last few years increased in popularity and obtained a good reputation for being an innovative and productive

way of conducting tests. The know-how of this method was secured when hiring the test coordinator.

The method chosen in D7.3 leaned towards a more step-by-step workflow and the time to perform the test was set three weeks after the functionalities were provided. That was the time needed to construct vast material to facilitate the testing sessions, construct entry and exit criteria for each test case and foresee risks and contingencies that might arise. This would in turn mean that the test result documentation would have demanded a lot more time to provide. Therefore a testing methodology where the preparations can be combined with the actual testing is preferred. The use cases are usually meant to be followed literally. However the use cases written for QVIZ had a slightly different character than the use cases to which the previous method was constructed. The test cases written to the QVIZ system kept a more descriptive form, which would have meant that all the use cases, user requirements and scenarios would have had to be reconstructed. The similarities to the methodology now chosen were many as well. Just as the exploratory method the decided workflow was to be disregarded at some point in order to more freely test the functionalities. Working close to the developers was recommended. However the most important reason for changing the method was the time limit as well as the concentration of quick feedback.

2.2.1 Exploratory testing

The chosen method for testing and validating the software of the QVIZ integrated platform is a method called Exploratory Testing. It is a method defined by a group of test methodologists now calling themselves Context Driven School. The popularity and the demand for Exploratory Testing methods have increased along with the development of agile development methods, characterized by flexibility in design and rapidly evolving functionality.

The simplest definition of exploratory testing is: “Test design and test execution made at the same time”, where the outcome of the most recent test will guide the tester towards the next. This means the testers are simultaneously learning the system, designing and executing tests, thus validating the software. Exploratory testing is even more suitable when requirements and specifications are incomplete or if there is a shortage of time.

The method is designed and optimized to find bugs fast and to serve as a quick turn around feedback for developers and management.

A comparison can be made between the more freestyle *exploratory testing* and its antithesis - *scripted testing*. Scripted testing basically means that the test cases are designed in advance, including which steps to reproduce and a thorough listing of expected results. Scripted tests are performed by a tester, who compares the actual result with the expected result.

When performing exploratory testing, there are no exact expected results. It is the tester that critically investigates the accuracy of the results and whether the outcome is to be considered correct or not. This also facilitates for the testers to consider the results in the perspective of usability and to make independent judgments as to whether this is the ideal solution or if there are additional features that may further enhance the functionality of the product.

The tests are set up based on pre designed Test Charters, which constitutes the basis for the test sessions. Test Charters include the purpose of the test, what feature is to be tested, data set up, expected results and the test intentions. The expected outcome and the expected results of the tests are on a higher level and

less instructive than for scripted testing, thus encouraging the testers to intellectually explore the system, as are the description of the test activities.

Everything the testers do is optimized to find bugs fast, so plans often change as testers learn more about the product and its weaknesses. Since the nature of the tests are exploratory and on a low level of detail, it is most crucial to take careful notes about the steps taken when testing and issues that were detected during testing. These notes then serve as a starting point for the next test session and retest of bugs as well as regression testing.

A recommended way of conducting exploratory testing is the so-called *session based testing,* which was developed in 2000 by Jonathan and James Bach. It is a method, which allows the testers to spend an uninterrupted period of time testing. Each session is focused on one charter, but testers can also explore new opportunities or issues during this time.

Session-based testing can be used to introduce measurement and control into an immature test process, and can form a foundation for significant improvements in productivity and error detection. This mode of testing is ideal when formal requirements are not present, incomplete, or rapidly changing. Usually two testers are testing and executing the test cases together; one performs them, while the other documents them. Session based testing is a software test method that combines accountability and exploratory testing to provide rapid defect discovery, creative on-the-fly test design and management control.

2.3 Exploratory Testing in QVIZ

Exploratory Testing is the most suitable testing method for this project mainly due to its flexibility and rapid feedback to management and developers. In addition to this the documentation and specifications of the requirements are written in a manner that is more suitable for exploratory testing.

The main reasons for choosing Exploratory Testing in the QVIZ project are:

- The method is optimized to find bugs fast providing a quick turn around of feedback to the developers in an interactive process.
- The functionality of the platform is rapidly evolving.
- The integration of the different components developed by the partners creates a dynamic setting for interpreting the usage of the features. Creative communication between partners and testers generate cooperative tools in the design process.
- There is a continuous change of data sets in the product as the source data constantly is being updated and improved. The archival as well as the geographical data is continuously changing and improving as the project evolves and the migration of data sets into the QVIZ platform progresses.
- The results of the tests and the issues raised by the testers provide meaningful reports to management for decision making on continuous activities and priorities.
- More time is spent on testing and less time on preparing documentation which enables the testers, as well as the development team, to focus on their most important tasks.
- The documentation and requirements of the QVIZ platform are written in a manner, which is less suitable for pre scripted testing. The specification

of use cases and usage scenarios are not ideal for preparing expected results in a way that is required for scripted testing to be rewarding and for the fulfillment of the objectives of software validation because of its mixture of usability and functionality objectives in the description of work.

This project applied a variation of freestyle exploratory testing that included some features from scripted testing, where the test documentation was pre designed and the expected results derived from the specification of use cases. The expected results were described in detail for applicable use cases, however on a level that still challenges and encourages the testers to independently analyze the appropriateness of the outcome. The tests were designed for the testers to thoroughly consider the feature at hand and explore all possibilities and variations in which it is supposed to work and where it might fail. This method allows testers to provide meaningful reports to management about the progress of the development while preserving the creativity that makes exploratory testing work because of its rapid feedback.

2.3.1 Session based testing

This project exercised both session based testing and regular non-session based testing. For the session based testing the testers teamed up in pairs, one person ran the tests and the other documented on a separate computer. The team set up encouraged the creativity of the testers as they complemented each other in vivid discussions about the functionality and results. Working in pairs made the testing efficient and very thorough as it stimulated the creativity of the testers and minimized the risks of any parts being overlooked the process or issues left unaddressed. It also created a comfortable test team that could work with separate functionality testing but still kept a close workmanship and frequently discussed and assisted each other in the different tasks.

2.3.2 Test organization

A test organization was set up separate from the development team to test functionalities but also to act as imaginary end users of the QVIZ integrated platform.

The testing team consisted of:

- Test Leader from NAE, responsible for Work Package 7 and member of Project Management Team.
- Test coordinator at HUMlab/UmU with previous experience as a software test leader.
- Professional tester with many years of experience in software validation, UmU.
- Representatives from the project with an insight in the QVIZ project. HUMlab/UmU, NAE, TID, SVAR.

The main part of the functionality testing was performed by project members at Umeå University. Having the majority of the testers gathered in one location made the test activities easier to coordinate. This vouched for a closeness of the team, which made communication easy and fast, avoiding misinterpretations and unnecessary repetition of the same tests. The closeness of the team also served as a foundation for knowledge sharing and co-operation. The geographical closeness

to project management and development team leader also enhanced collaborative decision making on design and bug fixing issues.

Testing and the preparation of test documentation has also been conducted at The National Archives of Estonia, The National Archives of Sweden and Telefonica I+D This built a versatile test organization with creative input from participants and groups with various areas of expertise.

The whole testing team had regular Skype meetings once or twice a week during the entire testing period. The meetings concentrated on planning the workflows, dividing responsibilities and reporting the progress. For better information exchange with the developers, the test leader and test coordinator participated in the weekly Skype meetings of QVIZ technical development team.

2.3.3 Documentation

The testing cannot be successful without adequate documentation of the activities and results. Proper and detailed documentation is needed both for preparing the tests and delivering sufficient feedback to the development team. The following documents were produced to document the test process.

2.3.3.1 Test Charters

Starting Point for preparing the Test Charters have been the Use Cases defined in the D 4.1.3 System Specification and Requirement Report and the Key Usage Scenarios from D3.3 Domain Ontology Report.

The Test Charter describes *what* shall be tested, *how* to test it and what problems to look for. The charter is not a detailed step-by-step instruction, it has to encourage the tester to take own initiatives when testing and propose changes to the functionality and layout.

Test Charters include:

- Test ID – The id number and name of the test charter.
- Actor – The user role the tester is playing when testing.
- Purpose – Purpose of the test, including particular objectives supposed to be achieved.
- Expected results - Expected outcome when verifying the functions.
- Setup – What needs to be in place in order to start using the intended function. (Configuration or a start menu/login)
- Priority – Based on the priority of the requirement or other considerations.
- Reference – Use Case, Specifications, User manual or other source of information.
- Data – Input data needed to perform the test
- Activities – Test ideas, a description of what actions the tester should take. With an as good as possible coverage of the functionality, yet allowing and encouraging the tester to still be creative in performing the test. The mandatory reference to the specification of use cases and expected results is combined with the testers own imagination and own ideas of possible activities. The user manual for the particular prototype that is being tested is a helpful source when creating test ideas.

- Variations – Additional events that could occur. Overwriting data, miscellaneous activities etc.
- Comments - Things to pay special attention to. What is expected as an adequate/correct outcome.

Test Charters were produced prior to testing. They were reviewed by the test leader and project management to ensure that all the main features were covered by the documentation. The Charters were constantly updated as tests proceeded, the functionality of the product progressed and as the testers learned more about the product. Additional activities were added to cover all variations of each feature every time the tests were executed and new test ideas arouse.

The key usage scenarios lay the foundation for the testers to get into the mindset of an end user, both while testing and compiling the test charters. The key usage scenarios were tested separately from the use case testing as a preparatory step for the usability testing and Case Study, where they were more thoroughly tested. The testing was mainly performed in the Collaborative environment with the focus on user perspective and the needs of specific target groups rather than system functionality and performance. The scenarios laid the foundation for the testers mindset when performing a user perspective test but also the grounds for compiling the test charters. The aim of the usage scenario testing was to make sure that the target user groups will be able to use the system in their research and that the QVIZ platform includes enough content (administrative units and archival resources) needed for the future usability testing. The Key Usage Scenarios will be further expanded and evaluated in the usability testing and case study.

2.3.3.2 Testing Notes

The testing notes record the test session and offer feedback to the developers. These include:

- Charter that was tested
- Test notes document version
- Name of tester/testers
- Date for testing
- Session Start and end time
- Geographical location of the test's execution
- Software version that was tested
- Detailed notes on how testing was conducted
- Comments on the activities from the charter
- Additional activities
- A list of bugs and issues (open questions, product or project concerns) found with applied Flyspray report ID numbers of the Bug Reports or Feature Requests
- Percentage of the extent to which the test charter was finalized
- Browser used to test the system

The notes have been carefully written during the testing sessions. This is done in order to be able to reuse them upon retesting and for the developers to be able to follow the actions that were taken and recreate the issues encountered. The notes were posted on the wiki for everyone to follow the progress of the testing and the issues detected by the testers. The testing notes served as a complementary foundation of test cases for each time the same function was tested. Each tester could carefully follow what the other testers had previously done and proceed

from there and use their ideas as a starting point/foundation for new ideas This created a valuable transparency of the process – that enabled all project members to be able to read the testing notes to follow the progress of the testing and to give feedback to testers and developers.

2.3.3.3 Document publication

A special test page, which is the core of the on-line diary, was created and added to the QVIZ internal wiki. The page comprises all the relevant documents, templates, guidelines and links to the other related wiki pages. The test page consists of:

- Links to the important documents, e.g. System Specification Report and Assessment Methodology deliverables
- Links to relevant internal wiki pages, for example P2a planning description and specification of use cases
- Link to the bug reporting site FlySpray (see below)
- Templates of test charters and testing notes
- Testing time plan, including dates and responsibilities
- Test documentation, test charters and testing notes

This page served as a guidance tool for the testers and many of the developers. Many of the documentations needed for testing were spread over a vast number of wiki pages. In order to avoid wasting precious time on searching for the documents this page gave the testers one page that summarized the documentation and links needed to perform the tests. The page was connected to WP 7 and could easily be found by any partner of the project. However this wiki page was more of a presentation of the activity within the functionality testing where finished products were uploaded.

Another tool was used in order for the test coordinator and the primary testers to safe-hold the valuable documentation, keep track of the testing versions and to keep some sort of distance to the developers. A project room was made available online for the ones that had documenting responsibilities were the version handling of the documents were facilitated. An online project room can just like a wiki have documents uploaded. The work can be structured into folders and the documents can be locked so that no one else can add information to the document while you are working on it. This way, many different versions of the same document can be avoided since it becomes apparent that someone is working on the locked document. This tool keeps track of the versions uploaded and one can easily go back to an older version since they are kept with different version numbers. The decision to put all word files onto this project room tool gave testers access to the original document files and it hindered the developers from being tempted to interfere with the written documentation. This since the documents that were posted on test page on the QVIZ wiki they were of a “read-only” character. The test charters and test notes were produced and then cleared by the test coordinator who sent them to a documentation coordinator that was responsible for putting the originals in the project room and pdf:s onto the wiki test page.

2.3.3.4 Flyspray

The test results were described in the test notes and from there the crucial issues were implemented onto Flyspray, where each bug or feature request reported was

given a number. This number was connected to the test note report, a connection was established to the vast documentation concerning the testing sessions and to the bug reports. Retesting was in this way facilitated when the tester could return to the bug report and see the status of the bug, read comments from the developers and other testers.

FlySpray administrates and facilitates bug reporting and feature request administration. It is a web based platform and an independent reporting tool. This served as a basis for communication about the test results between the test organization, development team and management and other participants on the project. FlySpray was chosen mainly for its web access and its cost efficiency.

Bug Reports and Feature Requests

A Bug Report is the result of a failed test and is reported when:

- A feature is not working according to specifications
- A feature is not working according to expectations or a according to the testers understanding of the functionality
- A part of a feature is not working
- Obvious GUI related mistakes are discovered
- Performance of the system is insufficient
- There is ambiguity in how to interpret the issue at hand, whether to treat it as a bug or a suggestion of improvement

A Feature Request is reported when:

- A tester discovers something in the system that is not considered a bug but the tester wants to raise the issue to management for investigation and possible measure.
- An existing feature could be done in a different manner which the tester considers would better serve its purpose
- Testers discover possible enhancements that concern usability or user-friendliness
- A feature that is neither specified nor implemented but is by the tester considered a desired feature in the system.

Reporting a bug/feature request included the following information:

Summary – Short description of the problem

Task type – Bug Report or Feature Request. If there is ambiguity the tester always reports it as a bug and leaves it to management to change task type if decided

Category – The component the report is concerning

Status – New/Unconfirmed until management has assigned valid status

Operating System – Browser used when testing

Severity – Severity of the problem

Critical – Stops further testing
High – Prevents from further testing of specific functionalities
Medium – Has large impact on the functionality
Low – Of little importance
Very Low – Merely esthetics

Priority – Management prioritizes the reports

Reported Version – Software version that the problem occurred in

Testers, developers and project management all signed up for accounts and could easily follow the reporting and the progress of the reported defects here called bugs and feature requests. These bugs were assigned bug numbers that was, as pointed out before, implemented into the test notes in order to facilitate retesting and follow up.

Testers had the privilege to open tasks, comment tasks and close them after fixes were made and retesting verified the solution. Issues detected during testing that were considered bugs in the system or feature requests for improved functionality or usability of the QVIZ platform were all entered into Fly Spray.

The Test coordinator and the development leader had accounts configured on admin level with all rights to perform actions on the reports. For each report filed into the reporting tool, management and test coordinator considered how severe the defect reported was, the importance of solving the problem, a possible priority and assignment. Bugs were assigned to responsible developer and project management for decision making and prioritizing. Status of the reports were updated continuously and closed when fixed and retested.

Developers could comment all reports but they were not allowed to close the bugs, since all bugs had to be retested in order to validate that the problem was solved before closing the bug report.

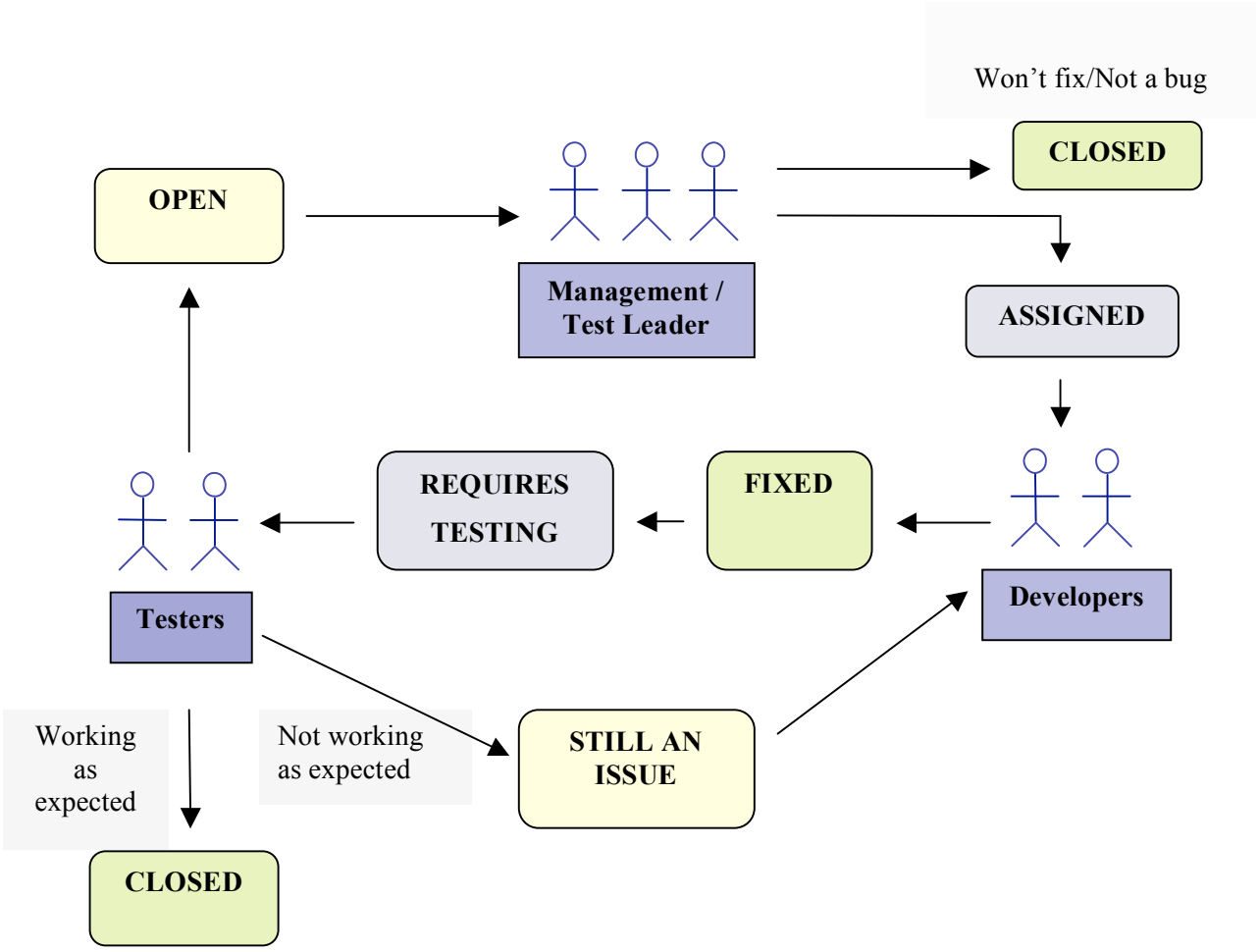

Figure 1. Roles and bug flow in QVIZ functionality testing

2.3.3.5 Bug reporting and role definitions

These are the role definitions and responsibilities in the test organization.

Test Leader

Create the test schedule for execution and divide the tasks between testers. Derive and develop the methodology of testing. Verified the correctness of the closed bugs

Tester

The tester constructs test charters founded on the use cases and the user manual. Then the tester executes the test charter on the application. After performed test they compile the test notes. Any bugs, feature requests or defects encountered during testing are added into the reporting tool Flyspray with supporting information so that the issue can be recreated by developers or for future retesting. Flyspray displays this reported defect as being in "Unconfirmed /New".

The tester who reported the issue on Flyspray receives e-mail notification whenever developer, management or any other tester makes changes or adds comments to the report.

The tester also receives an e-mail notification when a developer has mended the bug, or if management closed the reported bug.

A tester is free to re-open the bug and re-assign to the developer if the issue raised still is not functioning as expected. Flyspray then display this defect 'Reassigned'.

Management

Management receives an email-notification of the new issues reported in Flyspray. They then proceed to identify what the priority of this issue should have. The management assesses the validity of the issues, makes an initial sketch of priority and the severity of the issue at hand and assigns these issues to the appropriate developer. Because all reported bugs passed through development they could avoid that duplicates were created. This meant that they could add comments to the bug that already existed and close the new one.

The management's focal point was to assign the issues to the correct developer and Flyspray then displays the defect as "Assigned".

Developer

The developer receives a notification email of the reported issue and assesses the validity of the issue from their point of view. The developer acknowledges that there is an issue and begins to actively work on a solution. He implements the solution and becomes responsible for providing feedback to the tester regarding details of the problem and its solution. He is also responsible for editing the status of the report in Flyspray and to give an estimate on how many percent of the bug is fixed (this in order for the management to trace the progress). Flyspray now display the reported issue as 'Fixed / Require Testing' and sends an e-mail notification to the reporting tester for retesting.

The developer could be reassigned the bug if the issue still remains an issue and is not working as expected by the tester when retesting.

Bug/Error Life Cycle

The definition of a bug/error life cycle is: the duration between the time the bug is discovered ('New') to the time that the bug is closed successfully is.

During a bug's life cycle it is assigned various statuses, which are Open, Assigned, Fixed, Requires Testing, Reassigned, and Closed.

There are three different life cycles that a bug can experience:

Cycle I

1) A tester discovers a bug and reports it to Management (Development Leader / Test Leader).
2) Management verifies whether or not the bug is valid.
3) Management finds the bug not valid or not able to mend and the bug is therefore 'Closed'.

Cycle II

1) A tester discovers a bug and reports it to Management (Development Leader / Test Leader).
2) Management verifies whether or not the bug is valid.
3) The bug is verified and management reports this to the developer in question with the status 'Assigned'.

4) The developer mends the bug and marks the bug ‘Fixed’ and passes it on to the Tester for retesting.
5) The Tester / Test leader retests the bug and is working as planned, the tester then closes the bug and mark it ‘Closed’.

Cycle III
1) A tester discovers a bug and reports it to Management (Development Leader / Test Leader).
2) Management verifies whether or not the bug is valid.
3) The bug is verified and management reports this to the developer in question with the status ‘Assigned’.
4) The developer mends the bug and marks the bug ‘Fixed’ and passes it on to the Tester for retesting.
5) The tester retests the bug and the same problem remains. After confirmation from the test leader the tester reopens the bug and marks it ‘Reassigned’. The bug is returned to the developer.

By using this flow, all bugs are viewed by management and all bugs are finally closed.

2.3.3.6 Test Environments

Query Visualization Environment
The tests were performed in the order the features were developed and updated in the testing environment. The Prototype 2a http://polaris.regio.ee/qviz_p2a/ was delivered 2007-11-27. For Prototype 2b a test environment was set up immediately after the delivery of 2a at http://polaris.regio.ee/qviz_test/ and was continuously updated with bug fixes and new features to be tested.

Collaborative Environment
Prototype 2a for the Collaborative Environment was delivered 2007-12-03. Main part of the testing has been performed on this platform and the findings and results have served as guidelines for the next prototype. A test version of Prototype 2b was delivered 2008-01-28 and after this date continuously updated with features and bug fixes.

2.4 Results

Most of the tests were considered successful and most of the features were functioning well. The test effort focused on finding and identifying ill-working details, which is reflected in these results. It is not within the scope of this report to in detail present the features that passed the tests and were considered successfully implemented. This result section summarizes the issues identified by the testers as malfunctioning.

For an overview of the Use Cases and their implementation status see Appendix 1.

A list of the relevant reported issues and their status can be found in Appendix 2.

Here follows a summary of the findings from testing the P2a platform as well as a summary of the validation of the final platform P2b.

2.4.1 P2a

The P2a was the first fully integrated version of the QVIZ platform. The main objective of this deliverable was to integrate the different components into one platform. This integration was successful but at this stage the components

themselves in some ways had the characterization of prototypes, the main features were implemented but not yet fine-tuned and optimized.

2.4.1.1 Collaborative Environment Tool

Many main issues detected during testing of the first version of the collaborative environment were GUI related. Important features were difficult to access due to the sometimes complicated GUI, many functions were displayed but there was little or no information on how to access and use them. There was a lot of information in the environment, but it was not always relevant and sometimes even incorrect. The user manual did not provide the user with enough information.
Measures taken: The GUI was replaced in the P2b version and the majority of the issues were resolved and the user manual was substantially improved. Further improvements on the GUI can be made and is recommended.

Many of the functions were represented in text but they did not yet work properly and led to dead ends or were misleading as to what purpose they served. Membership handling was quite incomplete. There was no way to approve applications for restricted CoPs, yet users could apply. These malfunctions impaired navigation within the environment.
Measures taken: The main features were prioritized and they are functioning in the P2b version and additional features not fully implemented have been removed from the system.

The load indicator was inconsistent and caused confusion as to whether the page was stuck or still loading when performing certain actions. This problem was related to optimization problems. It made it hard to know what actually was a working process and what in fact was not working at all. There was no automatic refreshing of listings, which contributed to the confusion.
Measures taken: Significant optimization improvements were made in the final P2b version, both on the load indicator and the automatic refresh after a user updates information.

Confirmation and error messages when something was not working or the data added was incorrect are lacking in the system.
Measures taken: More confirmation on user activity is still desirable.

Language issues were detected. The environment had difficulties handling specific letters and presenting languages implemented in the environment. The system seemed to be confused by language parameters and caused loops of error messages that disabled users to proceed with navigation within the system.
Measures taken: Implementation if a different language handling system. Some issues still remain.

Metadata was generally presented in a complex manner, which made it difficult to understand the information.
Measures taken; Some work remains on the presentation.

Removal of unwanted items is not possible. There was no way to delete items such as Social Bookmarks from ones own folder.
Measures taken: The issue was resolved.

Means to communicate with other members and between users was lacking.
Measures taken: Moderators E-mail is displayed and there is a discussion feature to facilitate communication between users.

The solution using a default CoP that all users initially became members of had a presentation and usage that testers found confusing.
Measures taken: This is a remaining issue.

2.4.1.2 Map

The main problems with the map were related to optimization and performance. The map also had problems with an insufficient load indicator and locked the component if a user performed activities while the map was loading.
Measures taken: The map was considerably optimized in the P2b version. This issue is resolved.

The user manual lacked some crucial information on how to interact with the map.
Measures taken: The user manual was updated substantially with the P2b version

The map itself and it's correlation to the Faceted Browser had many complex features that provide the user with dynamic search methods and a lot of geographical information about the contents of the archives. This complexity sometimes made some features in the map slightly difficult to interpret, the time bar and the thematic layering in particular.
Measures taken: This issue is still a discussion amongst developers.

The presentation of the hierarchy of the administrative units were not clear, the different levels of borders were not distinctive enough to give a representation of the hierarchy and the relationship between the units.
Measures taken: Remains an issue.

The thematic layering sometimes covered the names of the map, the different levels on the zoom bar did not correspond to the actual zoom level and the background map did not always match the layering, which made it visually confusing at times
Measures taken: The thematic layering has been adapted to better show the background map and the place names on it. The levels on the zoom-bar are labeled with numbers and are less confusing. Issues were solved in P2b.

2.4.1.3 Faceted Browser

Optimization and performance were also issues with this component and information got lost on the way.
Measures taken: Enhanced in P2b, information loss enhanced?

Naming conventions were not coherent in the map and the facets, which confused the user.
Measures taken: The naming conventions in P2b are more coherent.

Some functions were not implemented completely and did not work in all cases, yet they were accessible, which was confusing.
Measures taken: The P2b version completed the intended features and removed the incomplete ones.

There were problems with the population and de-population of the facets when interacting in the browser. If the user activated an inactive facet, all information received so far was erased and one had to start all over again.
Measures taken: Sorted in the P2b version.

The faceted browser and map environment did not always collaborate as intended, information between the components was sometimes lacking and results were left out. For example when changing the time window the facets lost the information

gathered and the user had to choose facets terms as from the start of the search.
Measures taken: Many of these issues were sorted within the P2b version. Others remain a discussion and are still being developed.

2.4.1.4 Social Bookmarking

The social bookmarking feature had problems and was not working properly in the P2a version. It was unstable and it was often not possible to bookmark resources at all. These problems were mainly due to failure in communication between the different components.
Measures taken: The communication issues between the components (Archives and Collaborative Environment) were solved in P2b and are now working. It still experiences some instability in loading the information from the archives, mainly the Swedish National Archives.

2.4.2 P2b

Testing of the P2b version commenced in December for the Query and Visualization environment and in the end of January for the Collaborative environment. No issues reported after February 15 are included in this overview. Just before the release of the new version of the Collaborative environment, the consortium had a very productive meeting in Tartu, Estonia. The workshops during this meeting fed back a substantial amount of detailed and prioritized requests on the next prototype of the system.

These are the main issues and findings that were prioritized from the testers' perspective from the testing and validation of the final platform P2b.

2.4.2.1 Collaborative Environment Tool

Many issues were GUI related and recommendations to down size the environment by removing some functionalities and unnecessary information to get a leaner more user-friendly environment were given.
Measures taken: The developers implemented all but a few of the design related requests during the testing period of the P2b prototype.

The redesign of the GUI and downsizing of the environment resulted in the reemergence of some bugs that were previously reported and remedied in the P2a version.
Measures taken: Most of these issues were resolved also in the P2b version, but further improvements are recommended

One of the main problems encountered in the testing was instability of the system. The testers often needed to reload the environment and log back on, starting the testing session over again too many times. Also the Result area experienced instability problems. Right clicking on some objects in the CoP Navigator was not possible and sometimes the list moved around on mouse over, which made it impossible to select some of the items in the list.
Measures taken: These issues were resolved.

The search/filtering function did not work optimally. The searching was extremely slow and search/filtering on items related to a selected CoP resulted in a list of all of all items in the system.
Measures taken: These issues were improved and resolved.

The new prototype had problems with permission handling. It was possible to edit information in documents and communities that the user was not member of – even if this was supposed to be a restricted community. It was also possible to

delete important metadata from some items. Some things that should have been editable were not.
Measures taken: These issues were resolved.

The communication between the bookmarking application and the Collaborative environment experienced some problems. The bookmarks that were created were displayed twice in the CET and the page number of the resource that was bookmarked was not transferred properly to the CET.
Measures taken: These issues were resolved.

Some duplication issues were also detected. The CET seemed to save multiple copies of some items because it was possible to click the save button more than once. The primary topics entered by the users were duplicated in the list of secondary topics when listing CoPs in the search and result area. It was also possible to create several CoPs with identical names.
Measures taken: Work has been performed on fixing this problem however some of these issues still need to be resolved.

2.4.2.2 Map

The main problem for the map component was optimization and performance issues. Many of the bugs reported were directly or indirectly related to optimization.
Measures taken: The enhanced performance in P2b solved nearly all reported issues.

The time window did not always function properly as a filter. When moving the time bar, the information in the result list did not change.
Measures taken: This issue was resolved.

Another main issue was the thematic layering on the map. The thematic layering and borders sometimes hid the information on the background map, which made it difficult to read the geographical names and locating one self on the map. Also, when selecting an area on the map by clicking on it, the borders of the selected area was not highlighted.
Measures taken: These issues were resolved.

There was an issue about the interpretation of the information displayed on the map. Many of the maps features and thematic layering were not self-explanatory, which meant that a legend and more explanatory text were requested. There was also an issue with inconsistency in the use of language and labels in the map.
Measures taken: The map legend that was added is helpful, but further improvements are recommended, since it still is confusing for the testers to interpret the information given by it.

The map material was incomplete and was missing several cities in the north of Scandinavia as well as having an improper presentation of the areas where the background map is missing. For some time periods the administrative unit borders are dislocated from the borders of the background map.
Measures taken: These still remains an issues.

Finally there were some problems with the newly added function History of Point. This function also gave results when selecting a location in the sea and sometimes it also disabled the map and made the background map disappear as well as not working properly together with the time bar.
Measures taken: Most of these issues are resolved.

2.4.2.3 Faceted Browser

The facets were not populated with information automatically when selected, and if the user deactivated an empty facet this action erased all the information. Also if the user moved the dynamic timeline, the active facets got mixed up.
Measures taken: These issues were resolved.

The resources count in the indirect resources list was too high, sometimes even higher than the total number of resources. Resizing of the browser caused GUI problems such as the disappearance of the scroll bar.
Measures taken: These issues still need to be verified.

The facetted browser also suffered from the optimization issues for the map component. While the map was loading the whole browser was locked and unresponsive. This meant that the facetted browser did not work while the map was updating.
Measures taken: This issue was resolved.

2.4.2.4 Social Bookmarking

The new prototype also experienced some problems with the creation of social bookmarks and the communication with the other environments, primarily with the Collaborative environment. Since the P2b prototype for the Collaborative environment was implemented after the Query and Visualization environment, some changes to the bookmarking function needed to be made to adapt to the changes made in that environment.

Apart from the issues already reported under the P2a section there was also an issue with missing error messages when trying to create bookmarks without completing all the required fields. Sometimes the bookmarking icon was missing from within the archives and there are also some resources that are still not bookmarkable. The page numbers of a bookmarked resource got lost on its way to the collaborative environment and sometimes the metadata was not loaded into the social bookmarking at all. And when the bookmarks where displayed within the collaborative environment, it was displayed double.
Measures taken: Most of this was fixed in P2b, but a few of these issues still remain.

2.5 Obstacles during testing

Despite excellent communication between developers and testers some lacking in the response from developers were encountered. The features and functions had a fixed time schedule but kept being postponed and questions that needed answers in order to construct the test charters were delayed. The developers were under enormous pressure and did not prioritize the questions received by the test team. This was however solved by always having a representative from the test team at the development meetings. The two teams soon discovered the positive outcomes of this collaboration. However a lot of the testing time was spent trying to sort out what could be tested or not and this led to unnecessary delays for the testing team. Delays have been felt throughout the project and therefore even at testing. This was solved at a consortium meeting where the entire test team was gathered and set a timeline that was not dependent on the developers.

The open source tool Flyspray had bugs in itself and for a while the developers could close bugs. Since the responsible testers received e-mail notification of closed bugs, they could also make sure that the bugs closed by developers where re-tested and if needed re-opened.

2.6 Suggested actions and features for further implementation.

Mainly the suggested actions for further implementation and enhancement of the system is to focus on the features that today are in the system and fine-tune them to work impeccably and stable at all times. The test team also suggests that further implementation focuses on the usability issues regarding using faceted browsing and map component simultaneously when searching and filtering the archival resources, there are some usability issues that could be addressed to further improve the features.

The GUI of the Collaborative Environment Tool should be further enhanced. It is still complex and not very intuitive to use the features that are in the system. The layout still contains unnecessary and not very structured information that confuses the user and becomes an obstacle in the interaction with the system.

Manuals have been constructed for each deliverable of prototypes, however a lighter version should be constructed and posted online for the users to read.

Optimization of the system can be further improved to prevent people from having to wait while the system is loading and cause unnecessary irritation.

There are some Use Cases prioritized as Must that has not yet been implemented (see Appendix 1). Our recommendation is to complete these use cases only if the already existing ones are fine tuned and optimized.

The Usability tests will supersede the functionality testing and the results from that case study will be presented in the D 7.4 Case Study Evaluation report. This deliverable will give a good inkling to actions to take to perfect the system. The focus groups will be put together from the target groups. The test subjects that are involved in the usability testing will be of different ages, gender, computer knowledge, archival knowledge and system knowledge. The usability test will be performed both in Estonia and Sweden and will contain people from three different Universities. The test subjects will be given an assignment very much like a common archival question. To solve this assignment they will need to use all three main environments: The archival search, social bookmarking and collaborative environment. The results from these tests will give a roadmap to future implementation or continuation projects.

3. Validation of content

The content mentioned in the context of QVIZ regards information concerning administrative units and connections to resources within the archives. This content must be validated not only for the archives own sake but also in order for the test groups and focus groups to comprehend the information that can be found and explored within the archival environments. This validation of content has been made continuously. Not that many content bugs were reported into *Flyspray*, but many questions and data deficiency were discussed and sorted during meetings between QVIZ partners. Some of the detected bugs were known issues from the source data. Verifying archival data and complementing it, is an initiative of archives and has never been QVIZ's responsibility however not less important.

Datasets about administrative units have been provided and modified during the entire course of the project. Basic information concerning units and general structure of units together with sets of polygons were turned over to UoP, who has created an ontology of administrative units, the so called AUO, and forwarded the data to the time and space components developers.

Many inconsistencies became apparent when the map interface was implemented. The primary data issues, as well as problems with transformation of data from different sources were detected.

For example: some Estonian units, like parishes and manors, were added to map-component without containing proper dates. This created a situation where lower level units were shown as part of all higher level units at all times even if a lower level unit stopped existing many decades before the higher level unit was created. This was also reported as a bug in Flyspray (FS#229 - Relations between states and Europe). From the validations side it pointed out the necessity of providing dates for the administrative units.

Conclusions drawn from the validation of AU ontology was to make QVIZ time and space interface work properly and display information about AU accurately. The data from content providers should meet basic requirements:

1) All units must have start and end date – there were many flaws due to of missing dates. If exact date is not known (common to historical units), then rules for describing estimated date should be provided by QVIZ and those dates indicated by content providers.

2) Even if a more accurate timeline for units will be delivered in the continuous between QVIZ and archives, it is important that the data providers would be able to describe the levels of administrative units and types of units from the very start.

For archives it would be easier to describe their units and the units' structure if there were comprehensible rules constructed by QVIZ. Internal revision of the material must be encouraged and supported. It is also important to set up a regular update routine. From the archives point of view it is clear that data concerning historical units is never 100% complete – in most cases the resources were in some way insufficient.

Administrative units in Sweden where linked to resources by using geographical indexing system that is used by Swedish archives. In Estonia, archives do not have geographical indexing systems, NAE therefore linked their resources semi-automatically by using a program provided by NAE. The idea was to use names of

administrative units as words and matching/comparing them with all words in the name of resource. Matching led to a list of "direct resources". Indirect resources were connected to administrative units through QVIZ administrative units' ontology. This semiautomatic system helped to connect a vast amount of resources in short time period of time. Validation of those connections indicated that some follow up is needed to make system more reliable. For example some Estonian toponyms consists of words, which are relevant as nouns and because of that some administrative units were linked to nouns and not toponyms (exc. with parish name ""Risti" were linked materials about Red Cross (in Estonian "Punane Rist", "Kirjavahetus Eesti Punase Risti kaudu … (*letters to Red Cross…*)

4. Conclusion

The objectives set in the description of work stated that validation was important to:

- *Elaborate the QVIZ methodology for assessment and review of the project's result and progress towards the project's objectives*
- *Validate the software and specifications*
- *Assess the contextualization content to be included into the QVIZ-framework*
- *Create the project's Communities of Practice and build the community knowledge that enables effective validation and assessment of the system*

The use cases, the user scenarios and technical specifications that were developed in work packages two and four were based on the description of work and laid the foundation for the test charters. The connection to earlier work and project objectives is obvious and has provided a tool to keep to the initial objectives set by the consortium. The archives themselves have been eager to validate the content and make sure that there is valid and interesting information available in the QVIZ portal.

The new methodology chosen combined the test construction and reporting with the testing and providing feedback. Considering the many delays that this project has suffered from, this was the best way to construct the functionality testing. It created a steady flow of feedback from both testers and developers. This had however not been possible had not a professional test coordinator been tied to the project. Through the know-how and experience of these test team members the functionality test could be performed. Not only the experience but also the new view of things that they could provide was explicit. Problems and issues with communication, test performance and reporting could be foreseen and worked around.

The application of the methodology onto QVIZ testing worked well. The methodology was well equipped to handle a project like QVIZ. The fast feedback and the close connections between the developers and testers were preferable in a project like QVIZ where tasks are tangled and dependent on each other. In retrospect the professional testers should have been involved earlier on so that they had more time to get to know the system and give suggestions for improvement. The professional testers could have supported the project management with constructing a feasible time line and deadlines for providing functions. As it was now, the test management with the experienced testers had to learn the system, construct test documentation and start testing all in a short period of time. But by breaking free from the developments time schedule, the testing could be finished on time.

While documenting a test like QVIZ functionality testing, the handling of documentation is of great importance, especially when immediate feedback is crucial. A tool like Flyspray is vital for the communication between testers, management and developers.

Altogether a 100 use cases were specified in D 4.1.3 requirement list for development, implementation and validation. That list was based on revised use cases and a previous requirement list. These use cases were prioritized according to the MoSCoW method.

From the 100 Use Cases 46 were categorized as “MUST have this” (M), 44 as “SHOULD have this if at all possible” (S) and 10 as “COULD have this if it does not affect anything else” (C).
From use cases categorized as M, 41 (89 %) were implemented, 35 (80%) of the use cases categorized as S were implemented and from use cases categorized as C 3 (30%) were implemented.

Some of the Use Cases that were not implemented were fixed using other use cases or solved in a different manner than planned. For example use case QVIZ-UC62 “Removal of the temporal filter using "Remove Time Filter" button” can be achieved by using the widest extent of the time filter, which also yields the result that all administrative units and linked resources will be displayed.

The execution of the tests has proceeded according to the plans and there has been a great deal of improvements suggested by the testers. Most of these have been addressed by the development team and led to enhancement of the system. The remaining issues are mainly concerned with stability and usability of the system. The system has so many nice and unique features and information that is sometimes difficult to understand and interpret, which is unfortunate.

There has been some problems when conducting the tests related to lack of information from partners regarding development plans and dates of delivery of new features and corrected bugs. This sometimes led to difficulties in planning the test executions. The lack of detailed requirements sometimes led to a misunderstanding of the expected results, which for most times were solved after communicating the issues.

For further implementation the test team recommends that focus be given to improving the usability of existing features rather than adding new features and functionalities.

Appendix

A1. List of prioritized use cases and implementation status

These use cases have been prioritized according to the MoSCoW method.[1] That method provides a tool to prioritize and a better quality control for implementation and validation:

- M - MUST have this.
- S - SHOULD have this if at all possible.
- C - COULD have this if it does not affect anything else.
- W - WON'T have this time but WOULD like in the future.

The level of implementation of the use cases at the time of functionality testing has been rated with the following categorization:

- I – IMPLEMENTED meaning that this use case is implemented to an agreeable level
- PI – PARTLY IMPLEMENTED meaning that the use case is implemented to some degree, but not to the testers full satisfaction.
- NI – NOT IMPLEMENTED meaning that the use case has not yet been implemented by the developers, or that the implementation is not satisfactory.

This lists gives an overview of the use cases and their implementation status

No.	Description	Priority	Status
QVIZ-UC5	Get QVIZ account	M	I
QVIZ-UC60a	AU selection from Faceted Browser	M	I
QVIZ-UC60b	AU selection from map	M	I
QVIZ-UC61	Set temporal filter.	M	I
QVIZ-UC62	Removal of the temporal filter using "Remove Time Filter" button	M	NI
QVIZ-UC63	Change of the current year in the map using time slider	M	I
QVIZ-UC64	Facet filtering with map extent	M	NI
QVIZ-UC65	Removal of the extent filter	M	NI
QVIZ-UC66	Usage of the "History of the Point" tool	M	I
QVIZ-UC67a	Opening of the legend window. Visualize counting of archival resources on map for a particular time period and particular geographical area based on queries in facetted browser	M	I

[1] http://en.wikipedia.org/wiki/MoSCoW_Method

QVIZ-UC67b	Closing of the legend window	M	I
QVIZ-UC67c	Changing thematic coloring of the map	M	I
QVIZ-UC68	Selecting a polygon from chronological list of AU polygons in contextual area	M	I
QVIZ-UC69	Pan in map	M	I
QVIZ-UC70	Zooming in and out of the map	M	I
QVIZ-UC71	Swapping facets in the Faceted Browser	M	I
QVIZ-UC72	Add a facet in faceted browser	M	I
QVIZ-UC73	Remove facet in faceted browser	M	I
QVIZ-UC74	Select a CoP attribute in faceted browser	M	I
QVIZ-UC75	Selecting a resource reference in the result list	M	I
CoP-Mgt-UC21	List CoPs	M	I
CoP-Mgt-UC22	List users of a CoP	M	I
CoP-Mgt-UC23	View public information about a CoP member condition	M	I
CoP-Mgt-UC24	Get a CoP membership	M	I
CoP-Mgt-UC25	Create a new CoP	M	I
CoP-Mgt-UC26	View my CoPs	M	I
CoP-Mgt-UC27	Remove myself from CoP	M	I
CoP-Mgt-UC28	Remove member from a CoP	M	I
CoP-Mgt-UC29	Approve membership to a CoP	M	I
CoP-Mgt-UC30	Select active CoP	M	I
COP-COL-UC6	Create Tree based collection	S	I
COP-COL-UC24	View a social bookmark of archival resource	M	I
COP-COL-UC25	View a collaborative article from a collection	M	I
COP-COL-UC28	Add a social bookmark of archival resource to a collection	M	I
COP-COL-UC29	Remove a social bookmark of archival resource from a collection	S	I
COP-COL-UC30	Add a collaborative resource to a collection	M	I
COP-COL-UC31	Remove a collaborative resource from a collection	S	I
COP-COL-UC105	View collection hierarchies	S	I
COP-COL-UC106	View collection hierarchies of the User workspace	S	I
COP-COL-UC107	View collection hierarchies of the User's CoPs workspace	S	I
COP-COL-UC108	Paste a reference into a workspace collections	S	I

CoP-QVIZ-UC2	Store bookmark for archive resource sent from bookmarking web form	M	I
CoP-QVIZ-UC3	Query and Display bookmark(s) for archive resource from URL	M	I
CoP-QVIZ-UC4	Archive Hub - List of Users	S	I
CoP-QVIZ-UC5	Archive Hub - List of CoPs for User	M	I
CoP-QVIZ-UC6	Archive Hub - List of Users and their CoPs	S	I
CoP-QVIZ-UC7	Archive Hub - Store social bookmark for archive resource	M	I
CoP-QVIZ-UC8	Archive Hub - Store archive resource metadata	M	I
CoP-QVIZ-UC9	Show a list of social bookmarks of archival resources for any attribute of archival resource	S	I
CoP-UC1	Edit a collaborative page	S	I
CoP-UC2	Create a new typed collaborative article (page)	S	I
CoP-UC7	Remove protection of article	S	NI
CoP-UC8	Protect article	S	NI
CoP-UC51	Prepare wiki for publication	S	NI
CoP-UC61	Use WYSIWYG content editor	S	I
CoP-UC62	Manage important metadata for article	S	I
CoP-UC63	Create internal hyperlink in collaborative article	S	I
CoP-UC64	Create external hyperlink in collaborative article	S	I
CoP-UC65	Edit internal hyperlink within collaborative article	S	I
CoP-UQ66	Forward User to content editor when user selects open editing link for new article	S	I
CoP-UC100	Copy reference to clipboard for future pasting action	S	I
CoP-UC101	Paste reference copied within collaborative environment into content WYSIWYG editor	S	I
CoP-UC102	Simple Search in Collaborative environment	S	I
CoP-UC103	Display Result List	S	I
CoP-UC104	View public parts of any resource	S	I
CoP-UC52	Approve article for publication	S	NI
CoP-UC53	View public article	S	I

CoP-UC80	View semantic annotations of collaborative article	S	I
COP-UC81	View annotation of social bookmark of archival resource	M	I
COP-UC82	View annotation of link to content	M	NI
COP-UC83	Add annotation to collaborative page	S	I
COP-UC50	Assign a type to a collaborative page	S	I
COP-UC84	Remove annotation of collaborative page	S	I
COP-UC85	Add annotation to social bookmark of archival resource	M	I
COP-UC86	Remove annotation of social bookmark of archival resource	S	I
COP-UC87a	Add annotation to link to content	M	I
COP-UC60	Assign a type to an internal link in a collaborative page	S	NI
COP-UC87b	Remove annotation to link of content	S	NI
COP-UC24	Assign an Admin Unit (AU) property to a resource	M	NI
CoP-UC95	View social annotation for resource	S	I
COP-UC88	Add social annotation to resource	S	I
COP-UC96	Reply to social annotation	S	I
CoP-UC97	Modify social annotation	S	NI
Kn-UC-9a	Visualize Knowledge about Resource	S	NI
Kn-UC-9b	Visualize relationships between resources	C	NI
Kn-UC-9c	Manage ontologies and related resources	C	PI
Kn-UC-7	Manage a Schema	C	NI
Kn-UC-8a	Assign SKOs subject and primary subject to a resource	C	I
Kn-UC-8b	Add SKOS concept	C	NI
Kn-UC-8c	Create a SKOs based controlled vocabulary	C	NI
Kn-UC-8d	Manage a SKOs based controlled vocabulary.	C	NI
Kn-UC-8e	Navigate a SKOs based controlled vocabulary	C	NI
Kn-UC-10	View Folksonomy	C	I
Kn-UC-6	Discover knowledge source	C	I

A2. List of issues detected and their status

This list shows the relevant bugs that were reported during the Functional testing of the QVIZ prototypes.

Duplicates and bug reports that were not considered to be bugs have been removed from the list. On February 15 when the functional testing stopped the complete list of bug reports contained 218 items.

Note: Bugs fixed after February 22 are still labeled Not fixed in the list.

Bug Report	Reported issue	Severity	Fixed in version
	GENERAL ISSUES		
24	Clickable links should be visually distinguished	Request	Not fixed
	Content information is difficult to interpret	Request	Partly fixed
129	User manuals insufficient	Medium	Not fixed
	Error handling should be done better	Medium	Not fixed
	BOOKMARKING		
6	Unable to create bookmark	Severe	P2a
8	Tool tip in Bookmarking page is incorrect	User info	P2a
50	No CoPs to choose from when storing Bookmarks	Severe	P2b
52	Bookmark icon missing for some resources	Severe	P2b
62	Bookmarking does not work in IE	Severe	P2b
138	Resource metadata not always loaded	Medium	Partly fixed
170	Required fields and error messages not working	Severe	Not fixed
180	Page number of bookmarked resource not transferred to CET	Medium	P2b
	COLLABORATIVE ENVIRONMENT		
10	Link to user's homepage does not work	Severe	P2b
11	Extra title generated when creating any kind of annotations	Severe	P2b
12	Visualization tool is not working, it does not show the overview	Medium	Not fixed
13	Login page layout should be nicer and coherent with platform	Medium	Partly fixed
15,31	Create account page needs improvements	Severe	P2b
19	Inconsistency in having multiple windows open at the same time	Medium	P2b
28	Incorrect create date for CoPs and BMs	Medium	P2b
29	Sortable columns should be visually distinguished	Request	P2b
65	Language specific characters does not work in CET	Medium	Partly fixed
145	Not possible to accept/reject members/applicants of a	Severe	P2b

	CoP		
30, 150	Lists are not formatted to fit the window	Severe	P2b
130	Language handling in CET is corrupt	Medium	Partly fixed
103	Some CoPs not accessible in CoP navigator.	Medium	P2b
147	Insufficient restrictions on editing permissions	Severe	Partly fixed
118	The use and presentation of "CoP-QVIZ_Default_CoP" is confusing	Medium	Partly fixed
120	Freeze the labels in the lists so they are visible when scrolling	Esthetic	P2b
129	Insufficient information for users on how to interact with system	User info	Partly fixed
134	Irregular behavior of pop-up windows	Esthetic	P2b
137	Confirmation notifications missing after successful and failed actions	User info	Not fixed
139	Automatic refresh wanted when actions taken	Medium	P2b
146	CoPs a user is applicant is wanted in the list of my CoPs	Feature R	Not implemented
149	Too much and irrelevant information displayed	User info	Partly fixed
150	Information goes out of frame	Low	P2b
153	Metadata not presented properly	User info	Partly fixed
158	Should be easier to navigate within a CoP	Severe	Partly fixed
165	Contact information to moderator and members wanted	User info	Partly fixed
168	Removing BM from folder not possible	Medium	P2b
188	E- mail displayed with written words instead of signs "." & "@"	Esthetic	P2b
191	Right clicking in CoP navigator is unstable	Severe	P2b
192	Possible to delete metadata	Severe	P2b
196	Optimization needed for search/filtering	Medium	Partly fixed
197	CET creates duplicates since saving multiple times is possible	Medium	P2b
202	Not possible to display all bookmarks or archival resources	Severe	P2b
203	Primary topics repeated in secondary topics	Medium	Not fixed
205	Possible to create identical COPs, with the same name	Medium	Not fixed
212	Notification feature when new activities are made in CET	Request	Not implemented
217	Search/Filtering does not filter on chosen CoPs	Severe	P2b
164	View tab compound issues. Relevant information only wanted	User info	P2b

178	BMs displayed twice in the CET	User info	P2b
	ARCHIVE & CONTENT		
106,109	Naming of some AUs are not correct	Medium	Not fixed
110	AU history box contains confusing information	Medium	Not fixed
111	Double entries for some AUs	Medium	Partly fixed
206	IE problems in archive tools	Medium	Not fixed
	FACETED BROWSER		
4&5	Disable the buttons "flip page" & "swap" if the buttons are not applicable at that time	Esthetic	Not fixed
9	The tool tip covers the results in the Result list area	Esthetic	P2a
17	Replace abbreviations with understandable terms	User info	Partly fixed
20	Searching within facets should be easier	Medium	P2b
22	Feature to display Archival Resources on demand only	Request	Not fixed
27	AU history box contains swap signs for no reason	Esthetic	P2a
39	Data erased in facets when swapping	Medium	Not fixed
40	Disable or inform the user when not applicable facets	Feature R	Not fixed
45	Deactivating empty facets erases information in other facets	Medium	P2b
46	Verify translations and on-line text	User info	Not fixed
48	Inform the user when result list is empty	User info	Not fixed
59	Populate facets automatically upon activation	Medium	P2b
60	Map locks browser when loading	Severe	P2b
85	Direct resources are not always shown first	User info	P2b
86	Sorting alphabetically ÅÄÖ incorrect	Medium	Not fixed
128	Double clicking on facet name removes two facets	User info	Not fixed
136	Resizing windows after minimizing is a problem	Esthetic	P2b
159	Result list could be presented in a more logical manner	User info	Not fixed
177	Active facets get mixed up when moving the dynamic timeline	Medium	P2b
	MAP COMPONENT		
41	Inconsistent use of terms in FB/Map	User info	P2b
23	Feature to disable map on demand	Request	Not fixed
56	Map needs optimization	Severe	P2b
77	Time window does not filter results	Severe	P2b
78	Time slider is set incorrectly	Medium	P2b
83	Time slider sometimes locks the map	Medium	P2b

87	Map does not zoom to state level	Medium	P2b
91	Map layering cover names	Medium	P2b
93	It is difficult to interpret the map	Medium	Partly fixed in P2b
94	Load indicator insufficient	Medium	P2b
95	Activation of map (click), confuses AU selection	Medium	Partly fixed in P2b
100,101,102	Background map inaccurate	Medium	Not fixed
105	Feature to zoom on double clicking in map	Feature R	P2b
107	Zooming using keyboard sometimes disabled	Medium	Not fixed
115	Chosen AU from map is not highlighted with red border	Medium	P2b
116	Pink AU borders do not visualize the AU hierarchy	Medium	Not fixed
209	AU selection from map not working with IE	Medium	P2b
210	History of point gives result at sea	Medium	P2b
211	History of point disables the map at times	Severe	Not fixed
214	Map Legend insufficient	User info	Partly implemented