T-76.115 Project Plan: FASTAXON

Group: Muuntaja

Version History

Owner of the document: All members of the group Muuntaja.

Version	Date	Author(s)	Description
0.1	24.09.2003	Tero Leppänen	First Draft
0.2	29.09.2003	Tero Leppänen	Customer goals added, work practices updated
0.3	04.10.2003	Pekka Uusitalo	Added comments for some parts.
0.4	09.10.2003	Tero Leppänen	Updated according group comments
0.5	09.10.2003	Tero Leppänen	Updated according Pentti's comments
0.6	10.10.2003	Tero Leppänen	Customer goals added, communication and meeting practices updated
0.7	13.10.2003	Tero Leppänen	Action point register practice added
0.8	14.10.2003	Tero Leppänen	Chapters merged to master document after peer reviews
0.9	14.10.2003	Tero Leppänen	Pentti's personal assignment corrected: Design patterns
0.10	16.10.2003	Tero Leppänen	Test approach practice (PekkaU) added
0.11	16.10.2003	Tero Leppänen	Changes accepted (easier to read)
0.12	17.10.2003	Tero Leppänen	Risk Management added. Locked for review.
0.13	17.10.2003	Tero Leppänen	Page numbers added. Still valid for review
0.14	19.10.2003	Tero Leppänen	Updates after group review
0.15	19.10.2003	Tero Leppänen	More updates after review: corrections according Hannu's and PekkaU's notes. MS Word's spell check.
0.16	20.10.2003	Tero Leppänen	Phasing (I1) updated
0.17	20.10.2003	Tero Leppänen	Hannu's corrections to SCM chapter added
0.18	20.10.2003	Tero Leppänen	Link to Quality manual removed
0.19	21.10.2003	Tero Leppänen	Bug reporting chapter updated, Pentti's corrections to environments and skills
0.20	21.10.2003	Tero Leppänen	RedHat removed from tool list, task list table added.
0.21	21.10.2003	Tero Leppänen	I1 Task updated
0.22	23.10.2003	Tero Lepänen	Risk list added. Task list of PP phase added. Accepted by the customer.
1.0	26.10.2003	Esko Simpanen	Table 21 added. Table numbering unified. Styles, appearance and layout finalized for publishing.

Contents

1. Introduction

1.1 Purpose and scope of project

Purpose of FASTAXON project is to develop a system, which allows building, maintaining and browsing taxonomies by exploiting the faceted classification paradigm and Compound Terms Composition Algebra [1]. System should implement both administration and end user interfaces as well as database schema for FASTAXON.

FASTAXON is a part of research project in VTT. Main goal is to build a prototype system for demonstrating and proofing usability and flexibility of the Compound Terms Composition Algebra.

1.2 The system and environment

System contains several modules and user interfaces for them. Name of whole system developed is FASTAXON system. Interfaces may be either stand-alone applications or standard web-browser interfaces. Possible database runs in Windows 2000 server. In future database server may be changed to Linux. All stand-alone applications developed, must run in Windows environment. Some parts may include platform independent WWW-user interface. System is mainly mentioned to research use and it is users are technically oriented. However, a long-term goal is to adopt system to several application areas where information indexing is needed.

1.3 Rights to project outcome

The results (including source code) of the project will be published in VTT Open Source Server under VTT Public Licence Agreement [2][3].

1.4 Terminology and definitions

Table 1 shows basic terminology needed to understand this document. General vocabulary, terminology and definitions of domain are declared in [4]. More complete listing of terminology and definitions can be found in separate document listed in appendix 2.

Table 1. Terminology and definitions.

Term	Definition
Taxonomy	Hierarchy of terms
Compound Term Composition Algebra	Algebra for defining one meaningful compound term over a faceted taxonomy
MYOC	Merge Your Own code pattern
CCB	Change Control Board
SCM	Software Configuration Management
CVS	Concurrent Versioning System
RMB	Risk Management Board

2. Stakeholders and staffing

Project organization, stakeholders and roles are shown in figure 1.

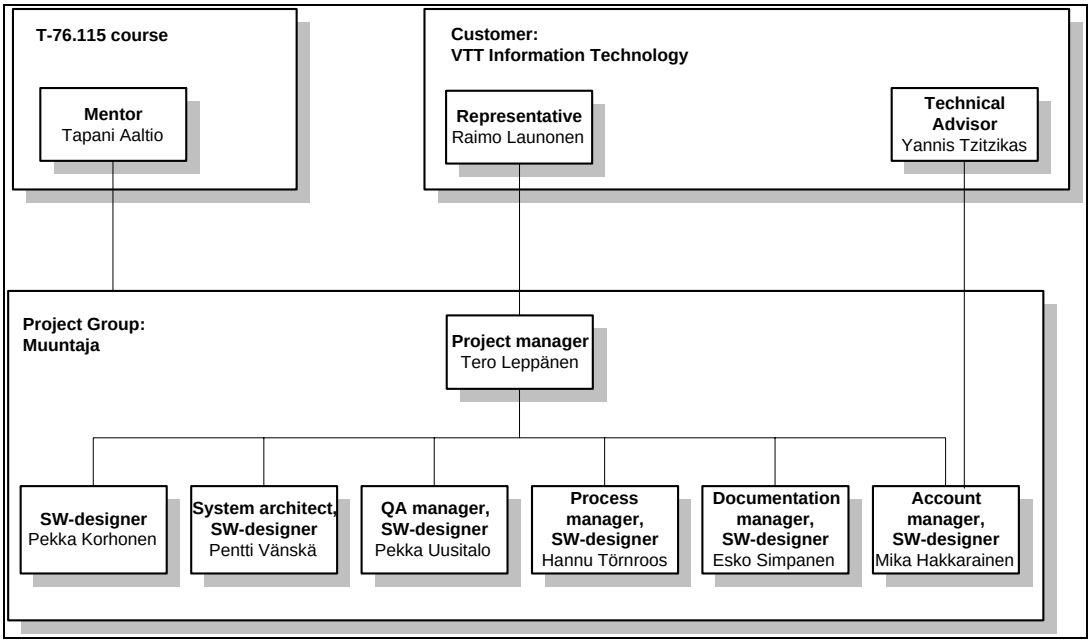

Figure 1. Project organization.

The project group Muuntaja is responsible of implementation of FASTAXON system. Muuntaja is lead by project manager. Account manager handles technical and requirement issues with customer's technical advisor and distributes information to project group. Mentor observers the progress of the project and advises Muuntaja group to accomplish project successfully and pass the T-76.115 course.

2.1 Project group

The project group consists professionals from IT sector. Members study in HUT in "Muuntokoulutus" –program for M.Sc. (computer Science) degree. Six members have B.Sc. degree and one has M.Sc. All the group members have a strong technical experience and work background in several software projects. Members are 3rd and 4th year students with 110-150 credits.

Table 2. Project group.

Name	Mika Hakkarainen
Role	Account manager, SW designer
Education	B.Sc. Electronics and Information technology, 1997
Major	Interactive Digital Media

Minor	
Work background	Research engineer in VTT Information technology
Programming skills	C/C++, Java in Windows OS (including PocketPC)
Other Skills	TCP/IP, project management, SW design, familiar with 3D-modelling
Name	Pekka Korhonen
Role	SW designer
Education	B.Sc., Construction technology, 1987
Major	Software Technique
Minor	Software Business
Work background	Develops Xsteed 3D-modelling software in Tekla Oyj
Programming skills	The main experience is with C/C++, but knows also Java
Other Skills	3D modeling, interested of programming and SW-design
Name	Tero Leppänen
Role	Project manager
Education	B.Sc. Electronics and Information technology, 1997
Major	Software Engineering and Business
Minor	Software Technique
Work background	Project manager and SW architect in Espotel Oy. Focused mainly on embedded systems.
Programming skills	Familiar with C and ASM, but knows also C++ and Java.
Other Skills	SW architectural design, communication protocols, requirements management and specification, UI design, project management, risk management
Name	Esko Simpanen
Role	Documentation manager, SW designer
Education	B.Sc. Electronics and Information technology, 1998
Major	Software Engineering and Business
Minor	The Venturing in Digital Economy
Work background	3D CAD and PDM specialist in CadWorks Oy.
Programming skills	Visual Basic, basic knowledge of Java and C/C++
Other Skills	API-programming, requirement management, and specification, UI design, testing, marketing and management.
Name	Hannu Törnroos
Role	Process manager, SW designer
Education	B.Sc. Electronics and Information technology, 1998
Major	Software Engineering and Business
Minor	
Work background	Test manager in TietoEnator Oyj
Programming skills	C/C++, Java, SQL.

Other Skills	Testing, configuration management, relational databases, project management.
Name	Pekka Uusitalo
Role	Quality manager, Test Manager, SW designer
Education	M.Sc. Biophysics, 1998
Major	Software Technique
Minor	
Work background	Software designer in Varian Medical Systems Finland Oy
Programming skills	Fortran, C, TCL/Tk, Pascal. Knows also C++ and Java
Other Skills	Quality expert, project management
Name	Pentti Vänskä
Role	System Architect, SW designer
Education	B.Sc. Electronic and Information technology, 1998
Major	Software Technique
Minor	Telecommunication
Work background	IT Specialist, Aerosystems Oy
Programming skills	Java, Perl, C/C++, ASM, Pascal, Fortran and Basic
Other Skills	Java application design and implementing with J2EE technology. TCP/IP, communication protocols, Linux, Unix, Windows, DOS, computer system administration, relational databases

2.1.1 Project group responsibilities

Responsibilities can be divided into two groups. First group contains general level project responsibilities, while another group contains responsibilities of individual software modules. Table 3 shows general responsibilities over the whole project.

Table 3. Responsibilities of the project.

Name	Responsibility area
Mika Hakkarainen	Requirement elicitation with the customer. Ensure that the remaining members of the group have sufficient understanding of the customer's needs throughout the project. SW engineering.
Pekka Korhonen	Bug reporting system (Bugzilla). SW engineering
Tero Leppänen	Project management, project meetings, project plan, progress reports, final report, and supervision of hour reporting system Trapoli. Member of CCB. Member of RMB.
Esko Simpanen	Documentation, document formats, deadlines of deliverables, project web site. SW engineering. Member of RMB.
Hannu Törnroos	Version control system (CVS). SW engineering. Member of RMB.
Pekka Uusitalo	Quality, testing. SW engineering. Member of CCB. Member of RMB.

Pentti Vänskä	Chief responsibility of the system's technical design, including architecture, interfaces, DB used and development tools. SW engineering. Member of CCB.

Technical responsibilities are assigned during design phases. Responsibilities to assign during design phases are for example: software modules, designer UI, navigator tree generator, end user UI, validity checker, storage manager and DB.

2.2 Other stakeholders

Table 4 shows stakeholders, their roles and responsibilities.

Table 4. Stakeholders.

Name	Role	Responsibility area
Raimo Launonen	Customer	To control project issues from customer side, grading the course
Yannis Tzitzikas	Customer's technical advisor	To give technical information, to provide test data
Tapani Aaltio	Mentor	Provide feedback and advises to project group, grading the course

3. Goals and end criteria

This chapter describes goals on a general level from different perspectives. This chapter is not intended as an actual specification of the project. The requirements document serves that purpose, and can be referred to if deemed necessary.

3.1 Goals of the customer

This chapter contains the Top-13 goals of the customer in order of priority, accompanied by their verification criteria.

Table 5. Top-13 goals of the customer.

	Goal	Verification criteria
1	To have a system for building quickly very big taxonomies or Catalogues by exploiting (a) the faceted classification paradigm, and (b) the Compound Term Composition Algebra.	The system implements the requirements as they are specified in the Requirements document.
2	Implementation of Storage Manager, i.e. of a module for storing faceted taxonomies, algebraic expressions and object indices using a DBMS.	A complete and functional software module.
3	Efficient implementation of the algorithm CheckValid(s, e) where s is a compound term, and e is an algebraic expression Prerequisites: (1)	The correctness of this module will be tested on specific examples, e.g. on the examples that are given in [1]. The latency time of CheckValid(s, e) test environment PC should not exceed 0.1 sec in a faceted taxonomy of 1000 terms.
4	Sound and User-friendly Expression Builder: A module for formulating algebraic expressions over faceted taxonomies. It will use the Storage manager for storing and loading expressions. Prerequisites (1) and (2)	The correctness of the Expression Builder will be tested by checking whether it can decide correctly whether an expression formulated by the designer is well-formed or not. User-friendliness will be measured according to the following two objective verification criteria: The designer is able to define the parameters of an expression without typing them (e.g. by selecting them from lists). The expression builder does not allow to the designer to give a parameter that does not belong to the genuine compound terms of the current operation. (This would be very helpful for the designer as it will reduce the size of the lists and at the same time it will not let him to make any error).
5	Implementation of the Navigation Tree Generator i.e. of a module that takes as input an expression e and derives the corresponding	The correctness of this module will be tested on specific examples, e.g. on the examples that are given in [1].

	navigational tree. The navigation tree should be available for browsing of object-bases. Prerequisites: (1), (2),	
6	A dynamically generated Web-interface to the navigation tree should be also provided. Prerequisites: (1), (2), (4)	Implementation of Web-interface as specified
7	Designer IDE: A simple Graphical User Interface that will provide integrated access to the user-interfaces of the Storage Manager, Taxonomy Editor, Expression Builder, Navigation Tree Generator and Object Indexer. Prerequisites: (1), (2), (3), (4), (7), (8).	Implementation of integrated designer interface as specified
8	Object Indexer: module for creating/updating object-bases where an object-base is a set of objects indexed by compound terms. The user will use the navigation tree in order to select the desired compound term. The resulting object indices will be stored using the Storage Manager. Prerequisites: (1), (4)	The correctness of this module will be tested on specific examples, e.g. on the examples that are given in [1].
9	Graphical Taxonomy Editor: A module for creating and updating taxonomies. It will use the Storage Manager for storing and retrieving taxonomies. Prerequisites: (1)	Scenario that will be used for verification: (1) The user creates a new project and uses the taxonomy editor for defining a new taxonomy and then it closes the project, (2) the user opens the project and adds some terms, changes the ends of some edges, and saves again the project.
10	Modular software design	The design should allow replacing/ re-implementing any module of the system without having to update the internal functionality of the rest modules. The modules of the project will be defined during the technical architectural design, which will be done in I1 phase.
11	Importer/Exporter of XFML and XFML+CAMEL files Module for importing faceted taxonomies and object indices that are expressed in XFML Module for importing faceted taxonomies, expressions and object indices that are expressed in XFML+CAMEL Module for exporting faceted taxonomies and object indices in XFML format. Module for exporting faceted taxonomies, expressions and object indices in XFML+CAMEL format.	The correctness will be tested on real XFML, XFML+CAMEL files (e.g. on those that are found in [5] and [6]).
12	The customer would like to further develop the system	The software project will be published as an open source project before the end of the course.

		The experimental evaluation will start before the end of the course so as to allow us to publish them soon.
13	Publish the results in an international conference or journal	The system is functional early enough.

3.2 Goals of the project group

Table 6 shows the Top-5 goals of project group. Realizations of goals are evaluated at the end of each phase. Table 7 shows personal learning goals of project group members. Each group member should evaluate his personal learning goal after project.

Table 6. Top-5 goals of project group.

	Goal	Verification criteria
1.	Learn new skills	Grade of course ≥ 3
2.	Get credits for the degree	Pass course T-76.115
3.	Stay on schedule and work efficiently	Each review should be in schedule. Max. 190 working hours / person.
4	Satisfy customer	Documents delivered and approved by customer. Use cases implemented and result approved by customer.
5.	Have reputation	Results of project published in an international conference or journal and all members of group will appear as authors

Table 7. Personal learning goals.

Member	Personal learning goal
Tero Leppänen	To learn new skills and tools, efficient communication methods
Mika Hakkarainen	To learn and get more familiar with new methods and practices for project management
Pekka Korhonen	To learn SW development process as a whole, to learn about architectures and software design, to learn more Java
Esko Simpanen	Learn to implement new tools and practices in SW project
Hannu Törnroos	To learn new SE methods and get hands on experiences from work methods different from my work organization
Pekka Uusitalo	To learn in practice project management, project quality assurance, risk management and iterative project
Pentti Vänskä	To get more experience implementing Java applications, practices in SW project and to learn more about architectures and software design

There are no significant conflicts between customer, project group or personal goals.

3.3 Project abort criteria

Project will be aborted if more than two people are leaving the group and work cannot be reorganized.

3.4 Project end criteria

The project end criterion is set by course T-76.115. The course end date is 7th April 2004. The project will end on this day, at the latest. The project may end earlier, if the FASTAXON system is in state where all customer and course requirements are met. Meeting with all stakeholders and project group members is held to make sure that all parties are satisfied with the outcomes and feel that it is time to end the project.

4. Resources and budget

This chapter presents the planned resources for the project.

4.1 Personnel

Effort of project group members differs between project phases. Calendar lengths and content of iterations are described in chapter 6. Total hours per person are set by course T-76.115 and it is 190h. Table 8 Shows planned effort of each member. Planning of PP phase is based on realized hours so far. Effort for next iterations is focused during planning, which is done at the end of each round. Realized effort of each round is presented at iteration reviews.

Table 8. Planned effort - hours to be spent on project.

Phase	Tero Leppänen	Mika Hakkarainen	Pekka Korhonen	Esko Simpanen	Hannu Törnroos	Pekka Uusitalo	Pentti Vänskä	Total
PP	55	55	25	35	45	40	25	280
I1	30	35	40	30	35	30	40	240
I2	40	45	70	50	45	50	70	370
I3	30	35	35	45	35	50	35	265
DE	35	20	20	30	30	20	20	175
Total	**190**	**190**	**190**	**190**	**190**	**190**	**190**	**1330**

4.1.1 Restrictions

- Week 52: Reserved for Christmas holidays
- Customer's technical advisor, Yannis Tzitzikas is absent 2.11.2003 – 9.11.2003
- Week 8: Pekka Korhonen is absent

4.1.2 Hours for personal assignment

10-20h / person is used for personal assignment, which brings several methods to project. Depending of assignment, they are planned to present in different phases during the project. One member of group brings his assignment at a time. He will also train other members to use method. Training will take 0,5h – 1h / assignment, which takes total 3,5h – 7h / person. However, personal assignments should bring benefits to the project. Assignments are selected so that they give best possible advantages to the project. Phasing of personal assignments is shown in chapter 5.1.

4.2 Materials

Table 9 shows personal development environments of project group. Personal environments are available during whole project. Because of standard nature of environments, they are easily replaceable, if failures occur. Customer has possibility to offer development platform if needed.

Table 9. Personal development environments.

Member	Hardware	OS
Mika Hakkarainen	1. Best P3, 600MHz 2. Dell Latitude C800, P3, 700MHz 3. Dell P4 1,9GHz	1. Windows 2000 SP4 2. Windows 2000 SP4 3. Windows 2000 SP4
Pekka Korhonen	P3, 350MHz	Windows 2000 SP4
Tero Leppänen	1. PC Workstation, Celeron 800 MHz, 256MB 2. Compaq NX9010, P4, 2,6GHz	1. Windows XP Professional SP1 2. Windows XP Professional SP1
Esko Simpanen	PC Workstation, P3 800MHz, 1GB	Windows XP Professional SP1
Hannu Törnroos	IBM Laptop T21, P3, 850MHz, 256MB	Windows 2000 Professional SP4
Pekka Uusitalo	HP Kayak PC Workstation, 2 x 400 MHz, 512MB	Windows 2000 Professional SP4
Pentti Vänskä	IBM ThinkPad T30 1GB	Windows 2000 Professional SP4

4.3 Budget

Theoretical cost of project is presented in table 10. Work time cost of 80 € / hour is used as basis for calculations. For simplify, same work cost is used for all stakeholders. Project group members have fixed hours, 190h / person. Mentor's hours are taken from course T-76.115 presentation slides. Customers and technical advisors hour are estimated.

Table 10. Theoretical cost of project.

Stakeholder	Work type	Hours	Total hours	Cost	Total €
Project group	All	7 members * 190 h / member	1 330 h	80 € / hour	106 400 €
Customer	Meetings	20 * 2 h	40 h	80 € / hour	3 200 €
	Other	15 h	15 h	80 € / hour	1 200 €
Technical advisor	Meetings	25 * 2 h	50 h	80 € / hour	4 000 €
	Other	40 h	40 h	80 € / hour	3 200 €
Mentor	Meetings	10 * 1 h	10 h	80 € / hour	800 €
	Other	5 iterations * 6 h / iteration	30 h	80 € / hour	2 400 €
		Total Hours	1515h	Total cost	121 200 €

5. Work practices and tools

5.1 Practices

5.1.1 Test approach

This is a high-level test plan for the FASTAXON system. The purpose of the test plan is to make sure that validation and verification of the system is done properly and all parts of the system are tested. This plan describes the test methods used in different phases of the project. Detailed test plan will be written as a separate document in next iteration (I1). Test practices described here are applied to project with consideration together with customer. The aim of customer is to use 80% of implementation time for technical design and coding. 20% of implementation time is used for testing.

Before the implementation developers are encouraged to include the planned unit tests as a list with the design proposal. List of planned unit tests help the test writer in designing the formal testing. In our iterative development process formal testing should be done at the end of each iteration. Testing is focused to the features developed in the on going iteration. Also system level testing is done as early phase as possible. Some parts of the system tests like formal installation and stress testing will be done only during the last iteration when optimization of the program is meaningful. The cross-reference table in test plan will be updated after every test round to make sure that all parts of the product will be adequate tested. At the end of last iteration all tests should be tested and product should have no major discrepancies.

Main responsibility about the test management will be on the Quality Manager. Testing and test writing responsibilities will be shared between the group members. Coders will not test their own code, except engineering tests. Following testing procedures are used in this project.

5.1.1.1 Testing documents *(all project documents delivered to course or customer, user documentation, manuals)*

Purpose of document testing is to validate documents. Testing must be done before document can be accepted. Documents are tested in every phase of the project, and every document that is delivered to course or to the customer must be accepted.

Every document must have owner and acceptor. Testing is done by reviewing the document before it is accepted. Reviewers (one or several) must fill the review log. The review log describes who, what, when and how is review done and what are the found discrepancies. The reviewer also makes decision whether new review is needed or not. The owner of the document collects logs, makes corrections and (if new review is not needed) gives document and review logs to the acceptor. Acceptor must verify that every part of the document is reviewed by at least one reviewer. Accepted version is put aside in a version control system and everybody in the project group are informed about it.

5.1.1.2 Engineering testing *(debugging, unit tests, code review)*

Engineering tests are done by the developer. Purpose of engineering tests is to make sure that unit that is under creation fulfils the functional and quality requirements of a program and the code is well commented, documented and easily maintainable. Discrepancies found in engineering tests are generally not reported to the bug database. Only if bug is not fixed and will be present in a next build of a program the bug report is done.

Debugging is informal test that is done by the coder to verify that the code meets the requirements. Debug sessions are normally not documented. Sometimes the code can have commented test lines etc.

Unit test is done by the coder to verify that code meets requirements. *Unit test is always documented.* The test documents can be informal but test procedure should be documented in such level that tests can be easily repeated by another developer. Testing environment should be also record as detailed as possible.

Code review is usually done by other coder with the original coder. Code review is not mandatory to record but it's possible to fill a review log. Purpose of code review is to make sure that quality of code is adequate and more than one developer get to know the code.

Automated unit testing is done as a personal assignment of a one member of the project group. Tests are created to very limited and carefully selected part of the code during the implementation phases. Purpose is to efficiently verify the critical part of the code by adding the automated test procedure to the code.

Heuristic analysis is done to the graphical user interface (GUI) as a personal assignment of one member in a project group. Purpose is to verify the usability of the GUI using formal method.

5.1.1.3 Functional testing (formal testing of functionalities with planned input and output)

The base of the functionality testing is in use-cases and requirements derived from use-cases. Testing is pre-designed, input and output are defined and test environment is fixed. Purpose of the testing is to verify that software meets requirements. Tests will be written during the implementation to the test plan.

In functional testing every requirement must be tested. There will be cross-reference table indicating the requirements, test related to the requirement and result of the test. Tests are done cumulatively during iterations and once the requirement is verified it's no need to test it again in future builds unless related code or requirement is changed.

5.1.1.4 System testing (integration test, installation test, multi-user stress test, security test)

The purpose of system testing is to verify the system level requirements that are not tested in the functional tests. The goal of system testing is to make sure that program meets the performance requirements, is working correctly and is possible to install and maintain in different hardware/software environments.

Integration test is done to verify that different units of the system can communicate together correctly. This includes also communication with the operating system and network resources.

Installation test makes sure that all elements of the system can be installed and updated.

Stress tests are performed to make sure that system meets the performance requirements set to the system in different hardware and software environments.

Platform validation test is a short test procedure that can be run to the different software/hardware configurations. It goes trough the major functionalities and system level requirements. It is used to determine that system is installed and works correctly in current environment. It can be also used as an acceptance test when build/release is moved from development to the quality assurance.

5.1.1.5 Discrepancy tests (Test procedure for found bugs)

The bug reports are created when a bug is found from the program in some phase of a project. A separate procedure is defined for handling bugs. When the product is released all bugs should be ranked and all severe bugs should be fixed. All fixed bugs should be tested. Test log for discrepancy testing must be filled.

5.1.1.6 Peer test

The course has determined that the peer testing must be done by using Session-Based Exploratory Testing approach. In this approach predefined test plans are not done. However testers will receive instructions how to use the program. Also testers are guided to focus to the certain parts of the program. Test plan document will describe the focus areas for the peer testers. Peer testing will be done in two last iteration phases (I2 and I3).

5.1.1.7 Customer acceptance test

Customer acceptance test is a formal test created with the customer to make sure that customer requirements are fulfilled. The test is fixed in early state of the project. The customer will provide the test data and the test environment for this test.

5.1.2 Communication methods

E-mail, phone calls and personal meeting are used as communication methods between project group, customer and mentor. Group members should communicate directly to customer and mentor if needed, however project manager should be noted what is going on. Easiest way to keep project manager informed is use of CC-field in e-mail messages.

5.1.2.1 Daily Communication between group members

In addition to personal meetings, e-mails and phone calls Yahoo! Groups is used as daily communication channel between group members. Yahoo! Groups has web based user interface and it is easy to access. Yahoo! Groups offers following services: on-line chat, document archive, calendar, message posting and storing. A group called muuntaja is established and all group members have access to it. Yahoo! is not used as “official” document archive. It is used for fast distribution and information change. It is supposed that members of project group will read

Yahoo! messages at least once in a day. Language of Yahoo! messages may be finish or English. Note: security issues such as backup are not guaranteed by Yahoo!

5.1.2.2 Status reporting

If there are no meetings with stakeholders, project manager sends weekly status report to customer, mentor and project group by e-mail. For each iteration review detailed report of status and costs (hours) are prepared.

5.1.3 Software configuration management (SCM)

5.1.3.1 SCM organization

Following persons form projects change control board (CCB):

- Tero Leppänen
- Pentti Vänskä
- Pekka Uusitalo

CCB is responsible for following tasks:

- Handle bug reports
- Handle technical change requests
- Handle requirements changes
- Decision of new release / build
- Decide who is release manager
- Decide who is allowed to merge code

5.1.3.2 SCM tools

The used SCM tool is Concurrent Versioning System (CVS). For more info about it see chapter 5.2 Tools. CVS facilitates the storing of project artefacts to centralized repository where backups are taken on regular interval. In this case repository is located in HUT CS departments disk server (Niksula) where backups are taken every night. CVS saves all versions of the artefacts stored in repository. It assigns new version number to each version of the artefact committed in the repository. This way we can "return" the older version of the certain document/file by fetching it from CVS repository to needed version.

5.1.3.3 CVS repository structure

The CVS repository structure is stored under projects root folder named 'fastaxon'. Underlying hierarchy is described in 'index.txt' document, which is stored in repository's root folder. This index file is describes the meaning of each folder created in repository. Actual files are not included in the index. Thus index is updated only if folder hierarchy is changed.

5.1.3.4 Build/release baselines

Baselines are divided in two categories a) build baselines done inside the iteration and b) release baselines done in the end of iteration. These categories are handled in separate ways.

5.1.3.5 Release baselines

Release baseline consists of all deliverables of certain iteration. The release baseline creation procedure is following:

CCB makes decision to make release from build that has passed system testing and nominates one project group member as release manager for the release.

Release manager makes the release build.

Release Manager performs release platform test for the release.

If the release does not pass the test, the build is returned to development.

If the release passes the platform test all delivered artefacts related with the release are marked to belong to certain release by CVS tag operation. Tag for release is called as release number. Release number has following format:

fxr_X_Y , where

fxr = Fastaxon release

X = major version number

Y = minor version number

After release tagging, the files are copied to "/release/fxr_X_Y" folder in CVS. The hierarchy under each "fxr..." folder depends from delivered releases content.

Version numbering scheme for releases

Releases are numbered following manner:

- Major version number is incremented in major releases. Each increments deliverables form major release. Change in major version number resets the minor version number to zero.

- Minor version number may be incremented when minor release is delivered to customer. Minor release can be e. x. fix to major release.

5.1.5.6 Build baselines

Build baseline is created each time system build is delivered to system testing or when project documentation is delivered to customer for approval during the iteration.

Creating build baseline for document:

Only documents that are versioned are baselined. These documents are baselined after it has passed the quality assurance review. Reviewers and review methods are selected separately per document. When document is baselined, it gets new version number. A new folder named as *document name_x_y*, is created to store the baseline version of the document.

Folders name consists from parts described below:

Document name is the name of the baselined document,

x is the major version number and

y is the minor version number.

Ex. Project plan_1_0 is the folder name for project plan version 1.0 baseline

The folder is created in baseline documents "home" folder and the baseline version is copied in this folder and folder with its contents is added to CVS repository. Future updates are done normally in original document and the saved baseline document is left as is.

Creating build baseline for code:

CCB makes decision that system is ready to be tested and nominates one project group member as build manager for the build.

Build manager makes the build.

Build manager performs simple platform test for the build.

If build does not pass the test the build is returned to development.

If build passes the platform test all code related with build is marked to belong to certain build by CVS tag operation. Tag for build is called as build id. Build id has following format:

fxb_X_Y , where

fxb = Fastaxon build

X = increment number

Y = build number inside increment

Version numbering scheme for documents

Document versions are numbered following manner:

- Major version number changes only when document is accepted officially after formal review. Minor version number is reset to zero when major number is incremented.

- Minor version number may be incremented every time the document is modified. Minor version number is incremented as long as change in major version number resets it to zero.

Version numbering scheme for system builds

System builds are numbered following manner:

- Major version number tells the iteration in which the build is made.

- Minor version number is linearly incremented counter of builds inside each iteration.

Example:

fxb_1_1 is first build of first iteration.

5.1.3. 7 Procedures for changing baselines (procedures may slightly vary with each baseline)

Procedure below is modified copy from [7]. In order to change the baseline following procedure must be followed:

The change request related to change must be stored as change request document in to change request folder in CVS.

The change to baseline must be approved by CCB.

If the change implementation work estimate is more than 5 workdays or the change has great effect on products architecture or function, the change must be approved by customer.

Changes are implemented on version under development. Changes are done during normal development work on schedule of iteration that implements the changes.

5.1.3.8 Procedures for processing change requests and approvals

Procedure below is modified copy from [7].

Change reporting documentation

All change requests will be stored in change request folder in CVS and bug reports will be recorded using Bugzilla.

Change control process

Flow of control is displayed in the graph below

Main points about the process are:

Only well prepared requests and reports are presented to CCB to preserve time

Each change needs to be verified either by the board.

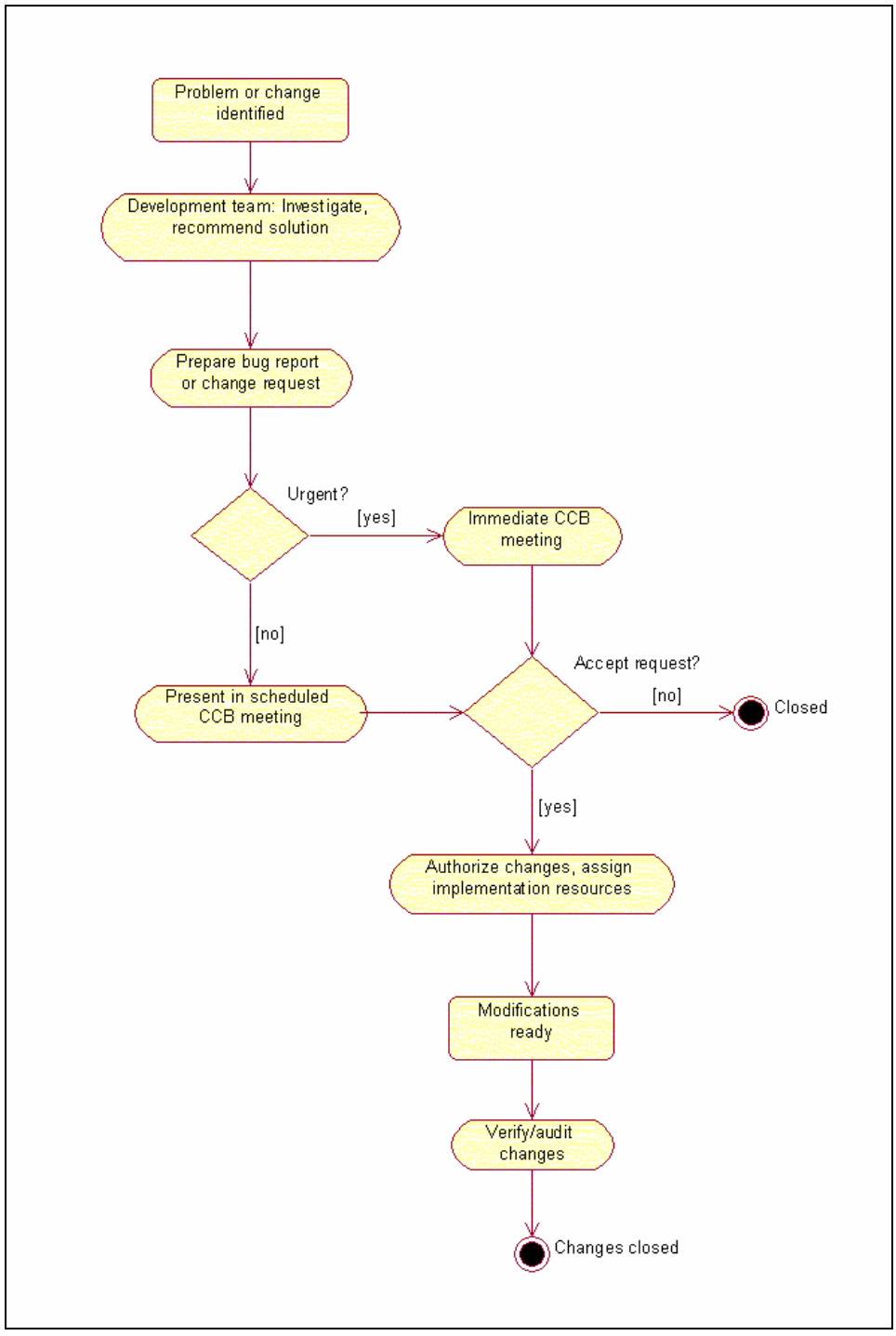

Figure 2: Flow diagram of the change control process from [7]

5.1.3.9 Branching and merging documents and code

To facilitate parallel SW development, following simple branching and merging methods are used.

5.1.3.10 Branching and merging documents:

Note: Method below is valid only if multiple project members are updating same document in parallel. If only one person is updating the document he may update the original document.

Each versional document has owner. See chapter 5.1.8 for details. Only the owner of the document may update the document. Thus if some other updates the document he will save the updated document to same folder with name formed as follows:

<document name>_<updater name>_<CVS version number of original document>.*

(* can be any extension which file can have)

The document under updating will be marked with draft status until update is finished. Then the updater will set status of the document as ready. Then the owner of the document will merge the changes to original document and moves the updaters version of the document to /old –folder located in same folder with original document.

5.1.3.11 Branching and merging code modules:

During the development the project structure is designed in a way that minimizes the need for parallel development. Anyhow some degree of branching and merging is required. Branch structuring is done manner similar to mainline branch-structuring pattern presented in [8].

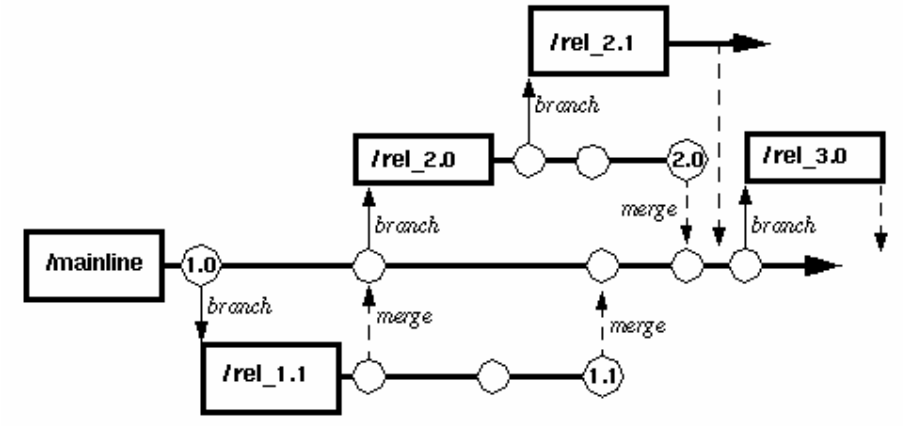

Figure S1b: Cascading Codelines with a Mainline

Figure 3: Mainline branch-structuring pattern from [8]

We maintain one main code line (mainline) that holds the whole stable system. Subsystem development is started in own code line and they will be merged to mainline when it is verified as stable. Thus branch elements are similar to Activity branch pattern presented in [8] and similarly branch creation corresponds to Branch per task pattern in [8].

The key point in merging is the used branching policy. Because the project group does not have very much experience in Java ™ coding we will start with very strict branching policy, where only CCB named persons are allowed to merge code to mainline. This way we achieve better code quality in the beginning of the project. This approach corresponds to Restricted-Access line branching policy pattern in [8] After group has got more experience from Java programming we may loosen the branching policy and allow coders to merge their own code to mainline. This approach corresponds to Merge Your Own Code (MYOC) pattern in [8].

5.1.4 Bug reporting, Bugzilla

In the project must be some system by which all bugs that are found in the product can be considered. Any bug that are found in testing, are assigned to CCB. CCB checks priority of the bug and assign it for fixing to the developer who is responsible for the module. Developer fix the bug and after own testing reassign the defect for verifying to the person who found the bug. He test that the fix works and if it does, he close the defect. If not, he reassigns the defect back to the developer. Only bugs that are found in official testing versions are considered via reporting system.

Bugzilla have been chosen for handling defects in the FASTAXON system. Bugzilla is web-based, open source bug reporting system. It allows adding of the defects to the database and assigning them to the responsible person. Any defect that has been added to the system can be found in Bugzilla by searching it, and after that, status of the bug can be checked. Bugzilla is also a good tool for estimating how much testing is required yet or is the product ready to releasing.

Following priorities used:

- Show stopper: Must be fixed immediately. Prevents releasing.
- High: Must be fixed before final version.
- Low: Should be fixed before final version.
- Cosmetic: Minor problems that don’t affect to functionality. For example small errors in GUI.
- Not a bug: It can be possible that although tester has found a “bug”, it is not the bug but user error. In this case developer can use this priority to tell it to the tester.

Bugzilla can be found http://www.soberit.hut.fi/T-76.115/03-04/raportointi/. Short instructions to use of Bugzilla :

Create new account:

1. Every person who is responsible for developing or testing the product must be create own account in Bugzilla.

2. Open the page http://www.soberit.hut.fi/T-76.115/03-04/raportointi.

3. Choose *New Account* and follow the instructions to give your user name and e-mail address to the system.

4. Password is send to you by e-mail, but your can change it afterwards

5. You are ready to use the system

Add a new defect to the system:

1. Login to the Bugzilla.

2. Choose *New*-button and *Enter Bug*-dialog is opened.

3. Type short and well descriptive string to *Summary*-field.

4. Give more information from the defect in *Description*-field. There you must describe the bug exactly and write step-by-step instruction how the bug can be reproduced. Set also priority here (Show stopper, High, Low, Cosmetic).

5. Fill all other fields. If you need more information choose the link to the help.

6. Write e-mail address of CBB to the *Assigned To*-field.

7. Choose *Commit*-button to commit the assignment.

Fix the defect:

1. Fix the defect that is assigned to you.

2. Test functionality of the fix and merge changes to CVS.

3. Login to the Bugzilla.

4. Find the defect in Bugzilla and open it.

5. Change status of the defect for “FIXED”.

6. Write additional comment “Fixed. Verify the fix, please”.

7. Assign the defect back to the person who found the bug.

Verify the defect:

1. Update your bin-branch from CVS.
2. Run the program and test carefully if the bug has been fixed or not.
3. Login to the Bugzilla.
4. If the defect had been fixed in the way you wanted, change status for "CLOSED".
5. If there was something to fix yet, change status for "REOPENED". After that, write additional comments for the developer and assign the defect back to him for fixing the problem again.

Find the defects:

1. Login to the Bugzilla.
2. If you know number of the defect, type it to the dialog field and choose *Find*-button.
3. If you don't know the number, you can search defects of the product by choosing *Quary*-option.

Some additional important notes:

- Use always descriptive string in Summary-field.
- Keep in mind that you describe the bug well and there are exact step-by-step instructions how the bug can be reproduced.
- Remember to mention number (or date) of the build in which the problem is occurred.
- Make sure that you are using the latest version of the product. If not, update your bin-directory and try to test again.
- Try to react to the assignment as soon as possible. Work of the many persons can be depend on you.
- Before you will add the new defect to the Bugzilla, make sure that it isn't here. It is not nice if there is a same bug in several defects. Of course duplicate defects can be merged afterwards.

More information from Bugzilla can be found in Bugzilla home page [9].

5.1.5 Time reporting, Trapoli

Each member of the group tracks the work effort they have invested in the project by reporting work hours to a Web based Trapoli system which is available at [9]. The Project Manager is responsible for administrating the project "muuntaja" in Trapoli. All the members of the project

use their own login usernames to enter data to the system. Hours are reported by task and work type. Tasks are created and updated by the project manager. The following fixed work types are categorized:

Table 11. Work types in Trapoli.

Category	Description
Project management	Project planning, tracking and control. Project reviews. Defining process and work practices. Creating substance for the project plan, progress reports, final report.
Design	Requirement specification and technical design, e.g., creating use cases, drawing design diagrams, specification of user interface and descriptions of interfaces. Architectural specifications, module interfaces and descriptions.
Programming	Creating, modifying, debugging and unit testing code. Low-level design done while writing code. Creating tools (e.g. scripts) for the project.
Pair programming	If the pair programming practice is used, the person responsible for the task reports his hours as programming. The other member of the pair reports his hours as pair programming.
Testing	Planning and executing testing (integration and system testing). Reporting defects. Other quality assurance practices, e.g., code reviews.
Documenting	Writing, reviewing and revisioning of documents concerning the software and the project. Time spent on brainwork creating the substance for a document before writing it in its final form should be reported to the corresponding work type, e.g., design.
Infrastructure	Installing and maintaining hardware and software tools for the project. Maintaining project web pages.
Meetings	Project status meetings, mentor meetings. Substance meetings on, e.g., requirements definition, design, etc. should be reported using the relevant work type for the topic.
Studying	Learning new project related issues (e.g. tools, technologies, standards, topic's domain).
Lectures	T-76.115 lectures

5.1.6 Meetings

There are tree basic meeting types:

1. Internal meetings of project group
2. Mentor meetings
3. Customer meetings

Project manager acts as chairman of meeting by default. Secretary for each meeting is selected among project group according alphabetical order of last name. Minutes of internal meetings and mentor meetings are stored in Yahoo Groups! files\minutes area. Minutes of customer meetings are stored to version control system.

Project manager prepare agenda for each meeting. Agendas of customer meetings are kept in version control system, where both customer and project group can access them. Agendas of internal meetings are distributed to project group via Yahoo Groups! Additionally project manager will send mentor meeting agenda to mentor by e-mail.

5.1.6.1 Weekly meetings

There are two timeslots for weekly meetings:

- Tuesday: 18:00-19:30 internal meeting, Computer Science Hall Cafeteria, Otaniemi, Espoo.

- Thursday: 16:00-18:00 meeting with customer, VTT, Tekniikantie 4B, Espoo

These meetings are held weekly if needed. Necessity of participation is agreed case by case, typically at previous meeting.

5.1.8 Documentation methods

The project generates a number of documents that are used as a communication tools between the stakeholders of the project. The table 12 below describes the main documentation of the project.

Table 12. Documentation.

Document name	Description	Format	Owner	Acceptor
Project Plan	Defines project level issues and practices.	MS Word	Tero Leppänen All members of the group Muuntaja.	Pekka Uusitalo
User Requirements	Describe the customer requirements of the FASTAXON software in general level.	MS Word	Mika Hakkarainen	Tero Leppänen
Technical Specification	Describe software requirements in detailed and technical level.	MS Word	Pentti Vänskä	Tero Leppänen
Testing plan	Defines the test cases used for FASTAXON.	MS Word	Pekka Uusitalo	Tero Leppänen
Test log	Used for bookkeeping test runs.	MS Excel	Pekka Uusitalo	Tero Leppänen
Test report	Used for reporting test runs against the test cases.		Pekka Uusitalo	Tero Leppänen
User's manual	FASTAXON end user’s manual.	MS Word	Esko Simpanen	Tero Leppänen
Action Point Register	Used for bookkeeping open tasks.	MS Excel	Tero Leppänen	Pekka Uusitalo
Progress report	Reports project status, practices and hours used.	MS PowerPoint	Tero Leppänen	Pekka Uusitalo

Terminology and Definitions	Vocabulary, terminology and definitions of the FASTAXON problem domain.	MS Word	Mika Hakkarainen	Tero Leppänen
Final Report	The final report of the project.	MS Word	Tero Leppänen	Pekka Uusitalo

In addition to these documents, separate documents will be created as needed for miscellaneous purposes. These includes agendas for customer meetings, minutes of the meetings, and installation instructions for the tools which are used etc.

Documents are stored into version management system using native tool formats. The final versions of the documents will be published on project's web site. PDF format will be used for publishing and delivering the documents to the stakeholders.

5.1.10 Action Point Register

Action point register is used for listing open tasks. Register is updated during meetings (internal, customer, mentor). Action point register is simple Excel sheet, containing tasks, their status and responsible person. Each task is named according current date and running number, i.e. AP20031013_1. Each task has Id, Description, responsible person, deadline, status and possible comments. Statuses of tasks are updated during meetings. Register is stored to version control system. Tasks listed in Action Point Register should be tiny enough so that responsible person can handle them. Each task should have only one responsible person; there are no shared responsibilities. If several persons are marked responsible, it means that all of them must handle task separately. For example if there task called "give comments about document X" and whole project group is marked responsible, it means that everybody should give comments of document X.

5.1.12 Personal SE assignments summary

Table 13 shows personal software engineering assignments required by course T-76.115.

Table 13. Personal SE assignments.

Practice	Responsible	Usage
Heuristic analysis	Mika Hakkarainen	I2-I3
Bug reporting, life cycle of bug	Pekka Korhonen	I3-DE
Communication methods	Tero Leppänen	I3-DE
Documentation methods	Esko Simpanen	PP-I1
Automated unit testing	Hannu Törnroos	I2-I3
Automated system testing	Pekka Uusitalo	I1-I3
Design patterns	Pentti Vänskä	I1-I2

5.2 Tools

This section describes all tools that project group will need on this project. This section describes also development and testing environments. Basically test environment is same than product environment. On this phase of the project it is not possible to know all the tools that we may need during the project. Main tools are basically clear. Some tools are also for customer use. On this point of view it is reasonable that all tools on product are free to use. This point is also important in the future after this project is over, because then it is much easier for the customer to continue further developing.

It is important that amount of the tools is as small as possible. This helps for example the maintaining. On the other hand, on software architecture's point of view, a simple construction is good. We must also remember that our resources are limited. On this phase it is clear that programming language will be Java. This system has tree main parts: client part, database and web server part. All programming will be done on Java. On database server there will be database management system software. On web server there will be web server software and some kind of servlet container. All the tools that will be used are described in tables 14-18. The development and the test environment are also described. These are preliminary valuations of platforms and tools. They are focused during project.

For avoiding overhead of configuration and learning new tools, brief installation manuals of each tool may be written if needed.

Table 14. Summary of tools needed.

Tool name	Version	System part	Use function	Note
Eclipse	2.1.1	All	Software developing	
J2SE	1.4.2_01	JRE in all parts	Software developing	
MySQL	4.0.15	Database server	Mandatory system software	this may change on next phase
Apache	2.0.47	Web server	Mandatory system software	
Tomcat	4.1.27-LE-jdk14	Web server	Mandatory system software	
Microsoft Office	2000		Documentation	
Dia	0.91		Documentation of UML diagrams	This may change during next phase
CVS	1.10 unix version		Version control of documentation and source files	Server
TortoiseCVS	1.4.5		Version control of documentation and source files	Front-end
Microsoft Windows	SP4	Clients and	Operation system	Also project

Tool name	Version	System part	Use function	Note
2000		database server		group operating system
Internet explorer	6.0	User interfaces	Platform	Compatibility with other browsers: to be defined
Bugzilla			Bug database	Web-browser front-end
Trapoli			Time reporting and project management	Web-browser front-end

Table 15. Development environment.

Software	
Tool	**Optional**
Microsoft Windows 2000	
Eclipse	
Java SDK	
MySQL	X
Apache	X
Tomcat	X
Microsoft Office 2000	
TortoiseCVS	
Dia	X
Hardware	
Processor	Intel P2 350MHz or better
System memory	≥256MB
Free disk space	≥1GB
Network connection	Mandatory for installations
CD ROM	

Table 16. Test environment for client configuration.

Software
Microsoft Windows 2000
Client version of FASTAXON application
JRE
Hardware

Software	
Processor	Intel P2 350MHz or better
System memory	≥128MB
Free disk space	≥10MB
Network connection	≥10Mbit/s
CD ROM	Optional

Table 17. Test environment for database server configuration.

Software	
Microsoft Windows 2000	
Database server version of FASTAXON application	
MySQL	
Hardware	
Processor	Intel 1GHz or better
System memory	≥512MB
Free disk space	≥1GB
Network connection	≥10Mbit/s
CD ROM	

Table 18. Test environment web server configuration.

Software	
Microsoft Windows 2000	
Web server version of FASTAXON application	
Apache	
Tomcat	
Hardware	
Processor	Intel 1GHz or better
System memory	≥512MB
Free disk space	≥1GB
Network connection	≥10Mbit/s
CD ROM	

5.3 Standards and guidelines

Following standards and guidelines are used in project:

- Code Conventions for the Java™ Programming Language Sun Microsystems [11]
- Java™ Look and Feel Design Guidelines, Sun Microsystems [12]

- UML [13]
- XFML [5]
- XFML + CAMEL [6]

6. Phasing

This chapter presents the phasing of the project. It lists the iterations' primary goals and deliverables, tasks and effort estimates, and critical dates during the iterations. The iterations follow T-76.115 course schedule.

At the end of each iteration, the next iteration's goals and deliverables are planned and documented here. If the plans change during the iteration, the changes must be presented at the project review in relation to the original plans.

6.1 Overview

Table 19. Project phasing.

Date	Iteration/Event
23.9. - 30.10.2003 (~4 weeks)	PROJECT PLANNING
27.10.2003	Delivery of documents and reporting
29.10.2003	Project review
31.10. - 4.12.2003 (5 weeks)	IMPLEMENTATION 1
1.12.2003	Delivery of documents and reporting
4.12.2003 (not approved)	Project review
5.12.2003 – 12.2.2004 (10 weeks)	IMPLEMENTATION 2
9.2.2003	Delivery of documents and reporting
12.2.2003 (not approved)	Project Review
13.2. – 18.3.2004 (5 weeks)	IMPLEMENTATION 3
15.3.2003	Delivery of documents and reporting
18.3.2003 (not approved)	Project Review
19.3.2004 - 7.4.2004 (3 weeks)	DELIVERY
5.4.2003	Delivery of documents and reporting
7.4.2003 (not approved)	Final demo

6.2 Project Planning

Goals:

- Project planning
- Understanding the domain
- Requirements specification on general level including most important use cases (business level use cases)
- Setup CVS-system
- Detailed planning of I1
- Define tools (preliminary)

Deliverables:

- Project plan
- Requirements document
- Progress report (slideshow)

Tasks:

- Table 20 shows tasks of PP phase and realized hours of them. Task "Write project plan" contains also actual project planning hours.

Table 20. Projectd Plan phase and realized working hours.

DS: Study GUI tools and technology	1.50
DS:Req. documentation	19.00
DS:Req. elicitation and analysis	4.00
DS:Study domain FASTAXON	9.50
GE: Meetings (customer)	42.90
GE: Meetings (internal)	17.50
GE:Lectures	41.50
GE:Meetings (status/mentor)	16.50
PM: Adopt tool CVS	8.50
PM:Other project management	7.50
PM:Personal SE practice	4.00
PM:Plan the next iteration	3.00
PM:Plan work methods and tools	8.00
PM:Project review and preparation	2.00
PM:Start and organize project	3.50
PM:Write the project plan	73.50
Project management	7.50

6.3 Implementation 1

Goals:

- To have defined and verified architecture for FASTAXON system
- Increase understanding of domain
- To have detailed technical specifications of system behaviour
- To have format (templates) for technical documents
- To have business-level class model

Deliverables:

- Architecture prototype
- Preliminary Database schema

Documents:

- Tech. specification (core architecture)
- Installation instructions for architecture modules and database
- Test case specifications (any format allowed, see the example from the testing guidelines)
- Test report of architecture prototype
- Updated requirements document
- Updated project plan
- Progress report (slideshow)

Table 21 shows I1 tasks, their responsibilities and planned hours. Biggest task is "Make prototype of selected architecture". It will be divided to smaller sub-tasks, after selection of architecture. Meetings are included inside tasks, except course meetings, which are fixed for whole project team. Gant-diagram of I1 is shown in appendix 1.

Table 21. Tasks.

	Mika	**PekkaK**	**Tero**	**Esko**	**Hannu**	**PekkaU**	**Pentti**	**Total**
Prepare architecture proposal(s) for customer	3	6		4			6	19
Select architecture with customer	2	2	2				2	8
Define preliminary database schema				2	4		2	8
Define test plan					2	7		9
Setup development environments	2	2	2	2	2	2	0	12
Make prototype of selected architecture		20					18	38
Setup test environment for architecture testing	4				2			6
Setup test data to database				6	2			8
Test architecture prototype		4				6		10
Verify architecture against business level use cases	4		2		4			10
Update requirements document	4		1					5
Update project plan	2		4					6
Make templates for technical documents				4		2		6
Prepare progress reporting and slideshow			8					8
Project management	2		8					10
Document management				6	2			8
Personal assignments	4	2			4	8	8	26
Course meetings (mentor + review)	2	2	2	2	2	2	2	14

Risk management			1	2		2		5
Release building for architecture proto		1				1	2	4
Platform test for architecture proto		1						1
Business class model	5				4			9
Class responsibility descriptions for business classes that have significant state dependency					1			1
GUI design / specification	1			2	6			9
Total	**35**	**40**	**30**	**30**	**35**	**30**	**40**	**240**

6.4 Implementation 2

Goals:

- To have first functional prototype of FASTAXON system fully fixed architecture
- To have most of the requirements labelled ‘must’ to be implemented
- Use case realization

6.5 Implementation 3

Goals:

- To have rest of the requirements labelled ‘must’ to be implemented
- To have as many as possible of the requirements labelled ‘should’ to be implemented
- If possible implement selected requirements labelled ‘optional’
- fully fixed requirements

7. Risk management plan

Purpose of the risk management plan is to define the risk management practices in faxtaxon project, evaluate the risks and plan the actions to minimize risks. Plan also defines who are the responsible persons and how the risks are monitored.

The main responsibility about risks management will be at the Quality Manager Pekka Uusitalo. He will attend to the Risk Management course (T-76.633) with the Project Manager Tero Leppänen, Process Manager Hannu Törnroos and Documentation Manager Esko Simpanen. This group form the risk management board (RMB).

7.1 Risk management practices

The RMB will handle risk identification, analysis, controlling and monitoring tasks. Part of this work is done in risk management course exercises.

Identification

RMB will have regular meetings once during each iteration phase for monitoring the risks. In these meetings the old risks will be re-evaluated and new risks are elicited. All members of the project group are encouraged to identify new risks and monitor old ones during whole project. If something alarming is noticed the project Manager should be informed as soon as possible. As part of the identification risk events will be defined.

Analysing

To each identified risk event probability and priority will be identified. The risks will be then ranked using the product of probability and priority. Corrective actions are defined for every risk event. Risk after corrective action will be analysed again with the same method. Corrective actions should drop the effect of the risk from critical or major to minor or small.

Table 22. Risk probabilities.

1	Highly probable
2	Probable
3	Possible
4	Unlikely
5	Highly unlikely

Table 23. Risk priorities.

1	Critical
2	Major
3	Medium
4	Minor
5	Trivial

Table 24. Risk effect table.

1 - 3	A = Critical risk
4 - 6	B = Major risk
7 - 12	C = Minor risk
13 - 25	D = Small risk

Control

Risks are defined in the risk table in the project plan. Risk event and controlling actions are also defined. Control of the risk is related to the risk effect table. Risks with critical or major initial

risk are assigned to responsible monitor who should monitor the risk and use corrective actions or inform the RMB if needed.

7.2 Risks

Table 25. Risks.

ID	Risk factors	Risk event	Effects	Initial risk	Controlling actions	Final risk	Responsible
R#1	1. The group is working while studuing. 2. Some group members have family with small children.	1-2 persons leave the project.	1. Too much work for rest of the group. 2. Some parts of the project become unknown to group. => The project is terminated.	B	1. Project manager will monitor the workload for individual members. 2. There should be a backup person for every task.	D	Project Manager
R#2	1. Project schedual is fixed by course 2. The group is working while studuing	Project deadlines are exceeded.	The group fail to pass the course	C	There will be responsible person for handling the course deliveries.	D	RMB
R#3	1. The group has limited knowledge of Jave coding 2. The theory behind the software is new	Coding is slow and project is delayed.	The group fails to meet the customer goals	B	The project scedual will be planned so that there is enough time to study the new tools.	C	Project Manager
R#4	1. The theory behind the software is new 2. The customer has limited tehcnical expertees	The software don't meet the requirements	The group fails to meet the customer goals	C	The progress of the project is monitored.	D	RBM

References

[1] Yannis Tzizikas, Anastasia Analyti, Nicolas Spyratos, Panos Costantopoulos. ”An Algebraic Approach for Specifying Compound Terms in Faceted Taxonomies”, EJC2003, Japan.

[2] VTT Public Licence Agreement

[3] VTT Open Source Server
http://opensource.erve.vtt.fi [referenced 12.10.2003]

[4] FASTAXON Terminology and Definitions.

[5] XFLM
http://www.xfml.org/ [referenced 19.10.2003]

[6] XFML + Camel
http://www.csi.forth.gr/~tzitzik/XFML+CAMEL/ [referenced 19.10.2003]

[7] Törnroos H, Kousa M, Naraneva L. 2002.
”CM Exercise plan for course T-76.614 Software configuration management”.

[8] Appleton B, Berczuk S, Caberera R, Orenstein R. 1998.
“Streamed lines: Branching Patterns for Parallel Software development”.
http://www.enteract.com/~bradapp/branching/references.html [referenced 19.10.2003]

[9] Bugzilla
http://www.bugzilla.org [referenced 21.10.2003]

[10] Trapoli, time reporting system
http://valinor.soberit.hut.fi:4288/trapoli_oht/ [referenced 19.10.2003]

[11] Code Conventions for the Java™ Programming Language, Sun Microsystems
http://java.sun.com/docs/codeconv/ [referenced 19.10.2003]

[12] Java™ Look and Feel Design Guidelines, Sun Microsystems
http://java.sun.com/products/jlf/ [referenced 19.10.2003]

[13] UML home page
http://www.uml.org/ [referenced 19.10.2003]

Appendices

1. PhaseI1Scheduling.pdf.

2. FASTAXONTerminologyAndDefinitions.pdf.