OMNIEDITOR PROJECT

PHASE B: PROJECT PLAN

CODENAME: NIRVANA

CSC408F1
Group Members:

Dylan Durdle
Yani Ioannou
Ashish Kaila
Sandy Lam
Gilles Lising
Mark Maglaque

CSC408H Phase B: Project Plan
Fall 2004

1 INTRODUCTION 2
1.1 Background 2
1.2 Aims 2
1.3 Key Personnel 2
2 PROCESS MODEL 2
2.1 Process Model 2
2. 2 Milestones and Deadline 3
2. 3 Critical Paths 3
3 PROJECT ORGANISATION 4
4 STANDARDS, GUIDELINES AND PROCEDURES 4
5 MANAGEMENT ACTIVITES 5
6 RISKS ANALYSIS 5
6.1 Lack of Common Understanding 5
6.2 Bad Planning 6
6.3 Loss of Key People 6
6.4 No Interest for the Project 7
6.5 Arguments amongst Team Members 7
6.6 Loss of Files 8
6.7 Bad Choice of Swap 8
6.8 Gold Plating 9
6.9 Workload from Other Courses 9
6.10 CVS Permissions 9
6.11 CDF Down 10
7 PROJECT ESTIMATES 10
7.1 COCOMO II Post-Architecture Model 10
6 STAFFING 15
8 METHODS AND TECHNIQUES 15
8.1 Technique 15
8.2 Tools 15
9 QUALITY ASSURANCE 16
10 WORK PACKAGES 16
11 RESOURCES 17
12 CHANGE MANAGEMENT 17
13 DELIVERY 17
13.1 Phase A Deliverables 17
13.2 Phase B Deliverables 18
13.3 Phase C Deliverables 18
REFERENCE 18
APPENDIX 19
Diagram 1 – Rapid Application Development 19
Diagram 2 – Gannt Chart 19

1 INTRODUCTION

1.1 Background

There exist many editors that programmers can use in the development of software applications. However, a programmer usually prefers coding using a particular editor and consequently, the time spent for learning how to use other editors can be more productively used in other tasks.

1.2 Aims

The OmniEditor package has been designed to bridge different editors through web services. This will improve the productivity of each programmer because it will allow programmers whose favorite editor is different from one's coworker, to work on the same file without any loss of data.

1.3 Key Personnel

- Gilles Lising
- Dylan Durdle
- Sandy Lam
- Ashish Kaila
- Mark Maglaque
- Yani Ioannou

2 PROCESS MODEL

2.1 Process Model

We will use the Rapid Application Development (RAD) method for this project. RAD allows usable software to be built in a short term period. This is particularly important since we have to complete 3 phases of

the project (refer to "Milestones and Deadline" below) within a short amount of time.

Please refer to Appendix - Rapid Application Development for a detailed process model of the project.

2. 2 Milestones and Deadline

The milestones and deadlines for this project are as per the course project schedule, that is :

Phase	**Deadline**
Understanding a Legacy System	Oct 7
Module Implementation Phase	Nov 4
Swap and Integration Phase	Dec 2

2. 3 Critical Paths

Critical paths are activities that has to be done at the date that was previously set in order to be able to complete the project. Below are the critical paths of the omnieditor project according to the activities shown in the Gannt Chart (see appendix):

Phase A

- Final project plan
- Final risk analysis report
- Final report of requirement specifications
- Final report of understanding of legacy editor
- Final test plan
- Final report of team organization
- Phase A submission

Phase B

- Implementation of load, download, insert, delete command
- Unit testing
- Deployment of web service and web service URL
- Final bug report
- Final maintenance plan
- Phase B submission

Phase C

- Pick a web service
- Implementation of client side
- Integration testing
- System testing
- Phase C submission

3 PROJECT ORGANISATION

The roles of each team member are the following:

- Project Manager – Gilles
- System Designer – Ashish (leader), Sandy (backup)
- Legacy editor analysts – Yani (leader), Dylan (backup)
- Testers – Gilles (leader), Mark (backup)
- Developers – All

Please refer to the Appendix - Gannt Chart for a more detailed description of each team member's responsibilities.

4 STANDARDS, GUIDELINES AND PROCEDURES

All the coding will abide by Java Standard Code Conventions, which is available at :
http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html

The text editor that will be used is Eclipse and the homepage for Eclipse is :
http://www.eclipse.org

The version control will be through CVS which can be found at :
https://www.cvshome.org

5 MANAGEMENT ACTIVITES

The following is a listing of the management activities that will be carried out during the lifetime of this project:

The *project manager* is responsible for:

- Scheduling deadlines for each task of the team
- Deciding the topics of discussion of each meeting
- Taking notes during meetings
- Monitoring the progress made by each sub-team in their task
- Motivating and supporting team members
- Emailing the team and updating them on the progress of the project

Test leaders are responsible for providing a test plan

The *legacy editor analyst leader* is responsible for providing the 'understanding the legacy editor' document.

Team members responsible for the requirement analysis must provide a requirement analysis document

6 RISKS ANALYSIS

6.1 Lack of Common Understanding

Risk :
- Misinterpretation of the project's requirements may involve partial or complete recoding of the software.

Prevention :
Each of the requirements should be discussed among all the members and in case of a disagreement, we should seek clarification near the T.A or the Professor.

Compensation :
In case of misinterpretation, try to figure out where changes have to be made and follow *Section 13 "Change Management"*

6.2 Bad Planning

Risk :
- Misestimating the level of difficulty.
- Inadequate time for testing.

Prevention :
Start the project early. Also, set an earlier deadline for ourselves so that if we come across some difficulties, we may ask for help.

Compensation :
In case that happens, try to seek help near the other members of the group.

Risk :
- Unreasonable schedule, such as a person has more work to do than the other members or unfair deadline.

Prevention :
The schedule should be fair and concise. Try to balance the amount of work among all the members and set the deadlines within a reasonable amount of time.

Compensation :
In case someone is unable to meet the deadline, he or she should let his or her teammates know well in advance so that appropriate actions can be taken. Also, if someone has been given more work than others, he or she should speak up.

6.3 Loss of Key People

Risk :
- A member may later decide to drop the course.
- A member may be unable to submit his or her part on time due to personal reason.

Prevention :
When assigning a particular job to somebody, another person should be chosen as a backup. The backup person should be aware of the work to be done and ensure that the designated person is doing his or her job.

Compensation :
The backup person will take over the job, try to work it out either by doing it himself or sharing the work among the others. Otherwise, the team should consult the professor as soon as possible to see what can be done.

6.4 No Interest for the Project

Risk :
- A team member may lose interest in the project.

Prevention :
Each member's ideas or opinions should be considered so that everyone feels part of the project.

Compensation :
In such a case, the PM or someone else should talk to the concern person and see what is wrong.

Risk :
- Other people may not be interested in our project during the swap.

Prevention :
The project should be multi-featured and at the same time should be simple to understand.

Compensation :
In that case, there is nothing the team can do. However, the team should analyze what went wrong and try not to repeat the same mistake with the swapped project.

6.5 Arguments amongst Team Members

Risk :
- Misunderstanding each other or differences in opinion may lead to disputes. Also, these arguments can prevent the project from being completed.

Prevention :
Everyone should be attentive at what his or her teammate has to say. Each team member should try and remain calm no matter how angry they might get.

Compensation :
In case of a dispute, the PM or another member should try to talk to both parties and settle things down. If an agreement can not be reached, the team should consult the course coordination to try and find a solution.

6.6 Loss of Files

Risk :
- Malfunctioning of a computer may result in loss of files.

Plans to minimize risk :
A backup of each document or source code should be made at least once a day.

Compensation :
Restore the backup which is only a day or less old.

6.7 Bad Choice of Swap

Risk :
- The swapped project may not match our expectations.

Prevention :
The team should carefully analyze other people's project and find the one that suite us best. Also, a thorough testing of the projects should be made so that we don't end up with a project full of bugs.

Compensation :
We should first test the software, debug it and try to add any feature that is not there, but we may require.

6.8 Gold Plating

Risk :
There is a risk that more functionalities than required are implemented.

Prevention :
Once the design is done, the team should go over the requirements again to make sure that there are no unnecessary functionalities.

Compensation :
If a needless functionality is found, it should be removed from the design or implementation as soon as possible.

6.9 Workload from Other Courses

Risk :
A team member may be burdened with assignments from other courses, which may lead to delays.

Prevention :
While scheduling the work, each member's workload from other courses should be considered.

Compensation :
If a member is overloaded with work, he or she should consult the other teammates to try and solve the problem.

6.10 CVS Permissions

Risk :
Wrongly setting the permission of a file might prevent some of the team members to view the file through CVS. Moreover, waiting for the party responsible to change the permission of the file may be a waste of time.

Prevention :
First, make sure that everyone knows how to set the permission properly. Also, someone should be responsible for doing a regular check if the permissions are correctly set.

Compensation :
If the problem of setting the permission arises often, we should find a solution to remediate to it. For instance, write a script that would automatically set the permission for each file entered in CVS.

6.11 CDF Down

Risk :
There is always the risk that CDF may be down thus preventing the whole team to work on the project.

Prevention :
There is nothing much we can do to prevent this from happening.

Compensation :
It is quite common that CDF goes down for a few minutes or even hours. Since we can not really do anything to prevent it, we should therefore anticipate that this event might occur. We should not wait till the last minute to do the work, especially everything having to do with CDF. We should set an earlier deadline for every work related to CDF, thus in case CDF is down, we still have time to work on the project later.

7 PROJECT ESTIMATES

In this section we try to provide estimates for the effort and time needed to complete this project. Since this is a short-term project, we can assume that we will not be acquiring more personnel.

7.1 COCOMO II Post-Architecture Model[1]

We will be utilizing this particular empirical model in our estimations since we have the system architecture devised from Phase A now defined and implemented. We have determined the source KLOC to be 4.190 (actual).

[1] Dr. Yijun Yu, "*Software Project Estimation"* in *CSC408 Lecture 7 Notes*, Slide 27. (Fall 2004).

Estimation of effort is calculated using the following formula:

$$E = a \times A \times KLOC^{b} \times \bullet_{i} costDriver_{i}$$

The values are determined as follows:

'a' is the type of project:

a	*Type of project*
2.4	organic
3.0	semi-detached
3.6	embedded

Figure 1: Three projects modes of software Development for the COCOMO II post architecture model[2]

Out of three possible choices we find that our project is organic because we have a relatively small team working on familiar environments and platforms. Most of our members have a very good understanding of how the current architecture would be able to interact with the user. We also share common skills and keep our communication channels open. Therefore, let a = 2.4.

'A' represents software reuse, calculated from the sum of weighted adjustment factors:[3]

A = 0.4 × fraction of the system requiring redesign (RD)
+ 0.3 × fraction of the system requiring recoding (RC)
+ 0.3 × fraction of the system requiring reintegration (RI)
+ software understanding increment (SU)
+ assess and assimilate reuse software increment (AA)

Since we have extensively planned out the design of the web service that part of the system would not really have to be redesigned. However, the system architecture of the legacy editor will need to be redesigned with the addition of new features and plug-ins in Phase C. Phase C represents about 57% of Phase B and C collectively, which are the two implementation phases of the project. Thus we estimate RD

[2] COCOMO Cost Estimating Web Site. "COCOMO Software Cost Model," Available: http://www.jsc.nasa.gov/bu2/COCOMO.html (November 7, 2004).
[3] Dr. Yijun Yu, "*Software Project Estimation"* in *CSC408 Lecture 7 Notes*, Slide 29. (Fall 2004).

to be 0.50. In other words around half of the system (the client-side part) will need to be redesigned.

The web service will not need to be recoded to specifications. On the other hand, the legacy editor (depending on our swap choice) may need some recoding although code would, more often than not, be added than rewritten. Thus, a reasonable estimate for RC would be 0.35.

For the reintegration factor the client-side of the system will need to be reintegrated together with the other part of the system (the web service) as it was not integrated before. We estimate RI to be 0.57, which is about what Phase C comprises of the two implementation phases.

We will have a choice of which editor we would like to develop for Phase C and so we would try to choose one that is quite modular and with sufficient documentation. Our estimate here is 0.20. In assessing and assimilating the reuse software, the effort put forth by our team would be for what is required in Phase C (i.e. extensive documenting, evaluating, and testing), and thus AA is 0.08.

Hence,
A = 0.4 (0.50) + 0.3 (0.35) + 0.3 (0.57) + 0.20 + 0.08
= 0.756

'b' is derived partly from a sum of five scale factors:[4]

$$b = 1.01 + 0.01 \times \sum_i W_i$$

Each factor W_i ranges from 5 (low) to 0 (high).

For each of the five factors we give the rating:

1.) Precedentedness: 1
We have the experience necessary to setup, develop, and deploy a web service within a reasonable amount of time and a very good understanding of the legacy editor architecture. Thus, it is not an

[4] JongMoon Baik, Dr. Barry Boehm, Dr. Ellis Horowitz, "COCOMO II Model Definition Manual," Available: http://www.dis.uniroma1.it/~santucci/IS/IS04-05/DISPENSE/CocomoII.pdf (November 7, 2004), p. 9.

entirely new project for us to develop. We also use ideas and concepts gained from previous courses.

2.) Developmental flexibility: 4
Although we have a choice on the legacy editor for Phase C, we are still restricted to developing code that needs to be in accordance with a web service interface designed by another team. Plus, we have had no client (or other team) involvement during the previous two phases but we will be working more closely than usual with the other team when developing the legacy editor in Phase C.

3.) Architecture risk/resolution: 2
We made sure that our interface follows from the given requirements for ease of use. Also, we should be prepared to handle even the most adverse situations as covered by our risk identification/resolution described in our risk analysis.

4.) Team cohesion: 3
Our team may have differing ways of going about the design and implementation of the project, yet as mentioned before we do share similar skills, keep communication channels open, and ensure that objectives are followed in a timely manner. Also, the first two phases did not involve other teams as part of our design and implementation and so inter-team cohesion may be an issue in Phase C.

5.) Process maturity: 3
We give this a rating of 3 because our system has not been tested outside our team and we do not yet have an accurate assessment of the bugs in the system not covered by our test cases. Yet we followed a preventative approach to reduce the occurrence of bugs in the first place.

So from the above estimates, $b = 1.01 + 0.01 (1 + 4 + 2 + 3 + 3)$
$= 1.14$

'$\bullet_i$ costDriver$_i$' *results from the product of seventeen effort multipliers:*

Attributes	*Selected Rating*
01.) Required Reliability	1.0 (Nominal)
02.) Database Size	1.19 (Very High)
03.) Product Complexity	1.0 (Nominal)
04.) Required Reusability	1.14 (High)
05.) Documentation Required	1.06 (High)
06.) Execution Time Constraints	1.0 (Nominal)
07.) Main Storage Constraint	1.0 (Nominal)
08.) Platform Volatility	1.15 (High)
09.) Analyst Capability	0.83 (High)
10.) Applications Experience	0.89 (High)
11.) Programmer Capability	0.87 (High)
12.) Personnel Continuity	1.10 (Low)
13.) Platform Experience	0.88 (High)
14.) Language and Tools Experience	0.91 (High)
15.) Use of Software Tools	0.86 (High)
16.) Multiple Site Development	0.92 (High)
17.) Required Development Schedule	1.00 (High)

Figure 2: Selected ratings of seventeen software development effort multipliers (cost- drivers) for the COCOMO II post architecture model[5]

Selection of these values and their explanations as applied to this project are based on the description of each of the ratings found via Table II-16 at the Parametric Estimating Handbook Web Site - http://www.ispa-cost.org/PEIWeb/ch6.htm.

Thus,

$$\bullet_i \text{ costDriver}_i = 1.0 \times 1.19 \times 1.0 \times 1.14 \times 1.06 \times 1.0 \times 1.0 \times 1.15 \times 0.83 \times 0.89 \times 0.87 \times 1.10 \times 0.88 \times 0.91 \times 0.86 \times 0.92 \times 1.0$$
$$\approx 0.74$$

We have all the values for the variables required to calculate E. Therefore,

$$E = 2.4 \times 0.756 \times 4.190^{1.14} \times 0.74$$
$$\approx 6.875 \text{ (total effort in man-months)}$$

Total project time T can be calculated using the following formula:[6]

[5] Parametric Estimating Handbook. "Parametric Estimating Handbook – Chapter 6," Available: http://www.ispa-cost.org/PEIWeb/ch6.htm (November 7, 2004).

$$T = 3.0 \times E^{0.33 + 0.2 \times (b - 1.01)}$$
$$= 3.0 \times 6.875^{(0.33 + 0.2 \times (1.14 - 1.01))}$$
$$= 3.0 \times 6.875^{0.356}$$
$$\approx 5.959 \text{ (total nominal project time)}$$

6 STAFFING

Please refer to appendix – gannt chart for more details on each team member's responsibilities

8 METHODS AND TECHNIQUES

8.1 Technique

Each developer will be responsible for writing and testing their own code. When the developer is done, he or she will hand the code to the tester who will thoroughly test the code using the black box testing method. Once the tester is satisfied, he will give the "go ahead" to the developer who is going to save it to the CVS while marking his or her name on the log.

Also, a debugger will be assigned and he or she will ensure that the code works as a whole. If an error occurs, the debugger is responsible for either fixing the problem, given that the error doesn't require a lot of recoding, or finding out the person responsible for the bug. The person responsible for the error will have to repeat the process again before making the change to CVS.

8.2 Tools

- Eclipse IDE
- CVS for source code version control
- Microsoft Office for documentation
- Open Workbench for gannt charts

---

[6] Dr. Yijun Yu, "*Software Project Estimation"* in *CSC408 Lecture 7 Notes*, Slide 30. (Fall 2004).

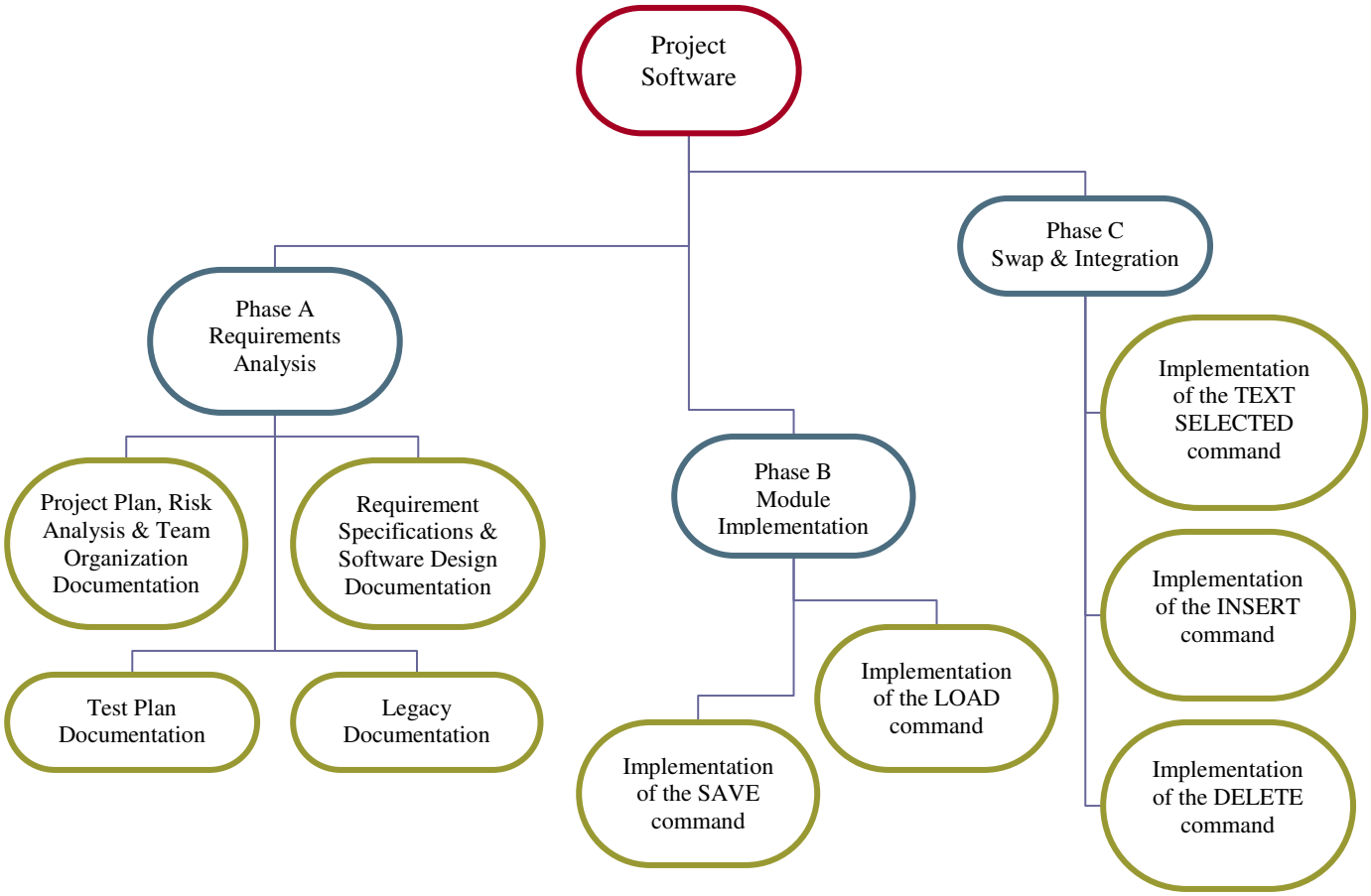
- MSN Messenger for online meetings
- Visio for diagrams such as Use Case and Sequence Diagram

9 QUALITY ASSURANCE

White box, black box, unit and integration testing will be performed to ensure a high quality software application. For more information, please refer to the test plan provided. Moreover, the testers will make sure that the software is meeting all the requirements in the requirement analysis document.

10 WORK PACKAGES

The figure below illustrates the different tasks to be performed at each stage of the project.



11 RESOURCES

- Apache Tomcat 3.2.x
- Axis1.1
- SAAJ API
- JAX-RPC API
- Jakarta-Commons Discovery
- Jakarta-Commons Logging
- Log4j
- IBM's WSDL4Java
- JAXP implementation
- Activation API

12 CHANGE MANAGEMENT

Any change that is required will be first discussed during one of our meetings, which is at least once a week. All the group members will analyze the change and see how it will affect the system. Once everybody is satisfied with the modifications to be made; the person originally responsible for that particular part of the system will have to adjust his or her code. If the modification requires a lot of recoding, one or more persons will be assigned to work together with the original developer to make the change.

The developers will have to abide by the "Methods and Techniques" section before a change can be saved to CVS.

13 DELIVERY

13.1 Phase A Deliverables

- Project plan & Risk Analysis
- Understanding of legacy system and the parts that can be reused
- Requirement analysis of the omnieditor and the selection of its architecture
- Test plan for the end product based on the requirement analysis
- Task allocation among members

13.2 Phase B Deliverables

- Source code of WSDL, server URL, web service implementation, unit test cases
- Documentation for user's guide, installation and deployment, traceability
- Requirements, Design and Implementation
- Bug report and Maintenance plan

13.3 Phase C Deliverables

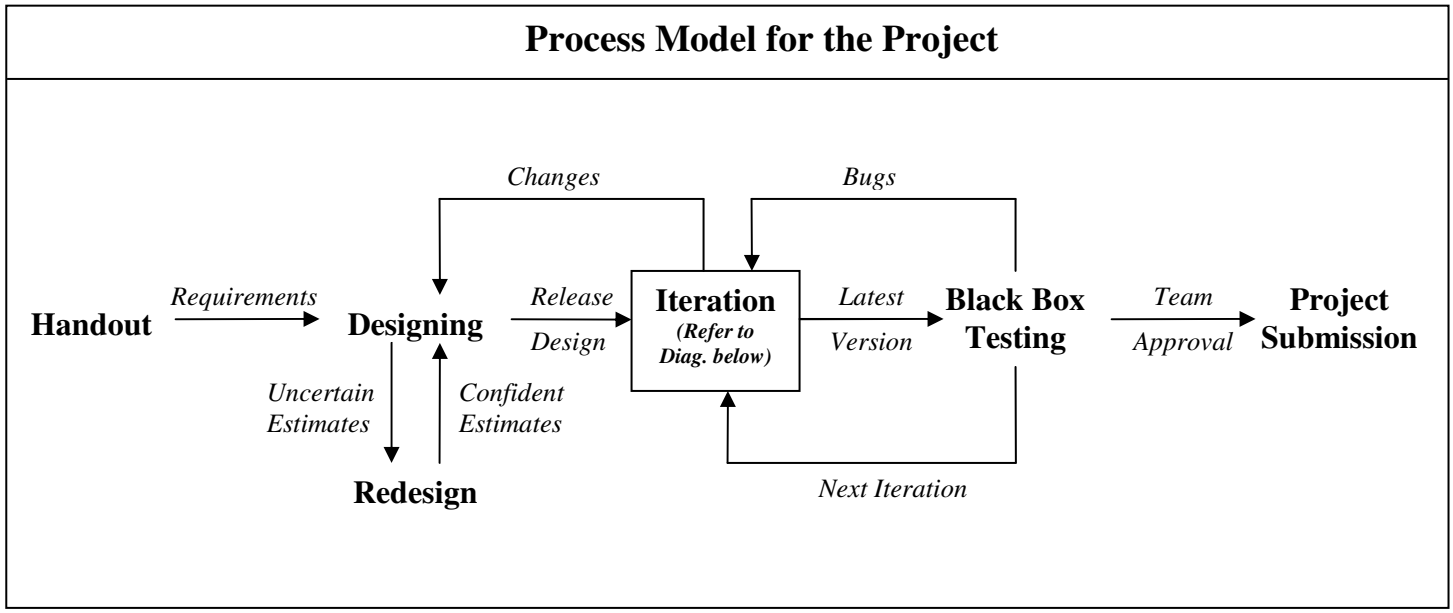
- Source code for an omnieditor editor client implementation
- Integration test scenarios
- Documentation for user's guide, installation and deployment, bug report and fixed bugs, measured software metrics during software evolution, optional

REFERENCE

- Software Testing – Diane Horton, Dept. of Comp. Sci U of T July 1999
- Government of Canada
- http://www.extremeprogramming.org/

APPENDIX

Diagram 1 – Rapid Application Development



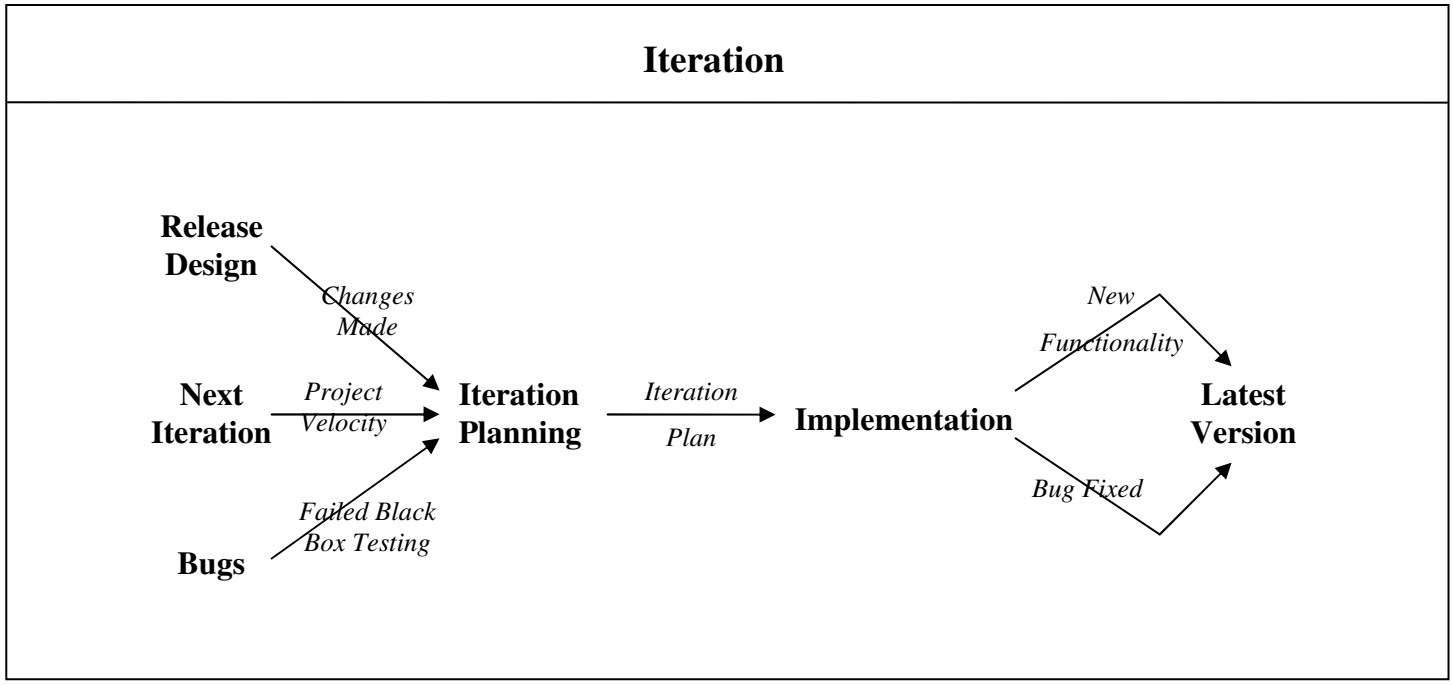


Diagram 2 – Gannt Chart