OMNIEDITOR PROJECT

PHASE B: MAINTENANCE PLAN

CODENAME: NIRVANA

CSC408F1
Group Members:

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

Table of Contents

Table of Contents 2
Introduction 3
Background 3
Aims 3
System 3
Personnel 3
Types of maintenance 4
Adaptive 4
Preventive 4
Corrective 4
Perfective 4
Pre-maintenance 5
Modularity 5
Structure 5
Documentation 5
Guidelines 5
Working tools 5
Pre-maintenance 6
Isolation 6
Post-maintenance 7
Backup Server 7
Modularity 7
Existing bugs 7
New bugs 7
Appendix 7
Appendix 8
Bug Report Form 8
Severity: Critical Major Minor 8
Bug Reproduced Yes No 8
Solution 8
Tested 8
Released 8
Maintainer's signature 8
Tester's signature 8

Introduction

Background

One of the major activities after delivering a product is maintenance. According to many studies maintenance is so essential that it covers around 50% of the costs of a product's lifetime. A maintenance plan has thus been developed by our team for this OmniEditor project.

Aims

At this stage of the OmniEditor project, our team is committed to providing a prompt and quality solution to any type of critical maintenance demands from the users of our web service. One of our main goals, in the pre-maintenance stage, is to reduce the need for performing any kind of maintenance activities during the Phase C of this project. In addition, procedures have also been set to deal with new bugs if any are discovered after the delivery of our web service.

System

Our deployed web service efficiently handles many operations from different users who want to edit one or more documents that were previously uploaded on our web server. The operations supported are document upload, download, browsing, line insertion, deletions plus concurrent document editing between many users among others.

Personnel

Maintenance involves an understanding of the core of the product. As a matter of fact, the persons most suitable are the developers who did the coding. For this reason, we have assigned our development team with the task of producing high quality patches since they are the ones who will know if any changes will have any ripple effects on other parts of the system.

Below are the roles of each team member:

- Dylan - Maintenance lead
- Mark - Maintenance assistant
- Yani - Maintenance assistant
- Ashish - Test lead
- Gilles - Test assistant
- Sandy - Test assistant

Types of maintenance

Adaptive

Since our product will only be used for about one month, we do not expect any changes in our users' environment in terms of upgrades and platforms issues. As a result, our team will not put a big emphasis on this type of maintenance in Phase C of this project.

Preventive

This type of maintenance has been performed in the pre-maintenance stage to help our team maintain our system with a minimum amount of effort. However, since the lifetime of this project is over after the completion of the course, preventive maintenance will not be our focus in Phase C of the OmniEditor project.

Corrective

Corrective maintenance is of significant importance and consequently, a lot of effort will be targeted at promptly fixing bugs for our web service users with a minimum of inconvenience.

Perfective

Since our web service will be used for about one month, this type of maintenance activity will be performed only if necessary to improve our system's performance.

Pre-maintenance

This stage of the software development is very important because it can significantly decrease the amount of maintenance that will result after the product delivery. For the OmniEditor project, we followed a strict set of guidelines to ensure that our users encounter a minimum amount of problems possible.

Modularity

Keeping our code modular will make our web service functionally more independent, making it easier for writing, testing and debugging. As a result, a lot of emphasis was put on high cohesion and low coupling during the implementation stage.

Structure

The structure of a code is a critical aspect of pre-maintenance. It improves readability and thus facilitates debugging tasks. For example, our development team has been following a no deeply 'nested if statements' and a no 'long procedures' coding principle and the same applies to our post-maintenance solutions.

Documentation

Very good documentation has been provided for each function including an accurate but brief explanation of the function. On top of that, preconditions and post-conditions have also been added to help our users in their client implementation.

Guidelines

A consistent naming convention has been used by adhering to the java standards. No ambiguous values are also being used. This results in better code, easier testing and debugging.

Working tools

In the software development phase of this project, our team used existing software tools that are known to be of very high quality. Using those tools minimizes the risk of involving any type of bug into our product. Simply put, we use high quality tools to ensure high quality products.

Pre-maintenance

Isolation

Each user will be provided with an isolated running backend service. If critical bugs are found and in the event of a system break down, it will not affect the other users.

Post-maintenance

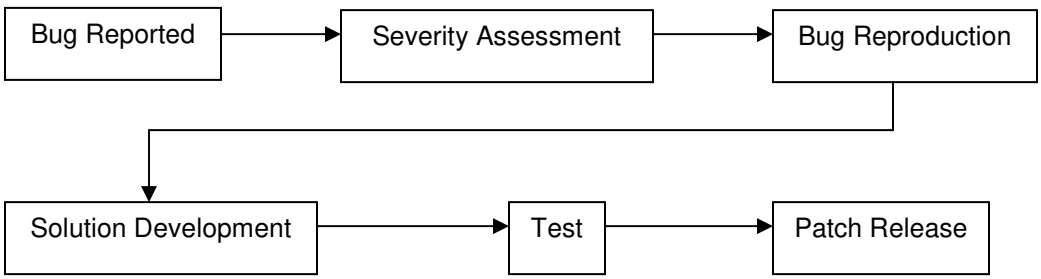
Backup Server
Our team possesses two additional servers that are already configured to efficiently redirect our users in the event of a CDF server breakdown.

Modularity
Modularity has to be kept if any big major addition is made to the source code for the same reasons as mentioned earlier in the pre-maintenance stage. It will simply assist in easier testing, coding and debugging.

Existing bugs
Any bugs left in our bug report will be resolved as soon as possible to ensure that the users of our product possess the best web service as deployed by us.

New bugs
All of our contacts such as each member's email address and MSN will be given to the users. Any bugs found can be reported and a form (see appendix) will be filled out with a bug assessment regarding its level of severity, one of 'critical', 'major' or 'minor'. Higher severity issues will be dealt with first and in a first-come-first-serve basis. Our maintainers will try to reproduce the same error and if successful, a solution will be developed. A selective testing approach will be performed to ensure that there are no negative repercussions on the rest of the system. Afterwards, the fix will be released to all of our users as soon as possible. If we were unable to reproduce the bug, the bug reporter will be notified to provide us with more details.

The diagram below illustrates the 6 steps involved.



Appendix

Bug Report Form

User #:	**Date:**
Description:	
Severity: (please check one box)	**Critical** ☐ **Major** ☐ **Minor** ☐
Bug Reproduced (please check one box)	**Yes** ☐ **No** ☐
Solution ☐	
Tested ☐	
Released ☐	
Maintainer's signature **Date:**	**Tester's signature** **Date:**