

Software Maintenance Plan

Patrick Flood, Lance Dubach, Sarah Althoff and Rudolph Strong

CSS 561 Programming Concepts

Yong Jiang

June 6, 2005

University of Phoenix

Abstract

Software Maintenance is a required aspect of any software development program. Software is almost useless if a way to maintain it is not established. This maintenance plan must be well understood by the developers and customers purchasing the software. The following paper will outline the importance of a maintenance plan, provide a sample plan, and process.

Software Maintenance Plan

Maintaining software is important for a few reasons. First, software and hardware constantly change. Therefore, it is important to keep applications synchronized with each other. An application's interface may need to be modified to incorporate new hardware or software features. Secondly, an application itself may require a new look and feel. Customer requirements change, therefore the application that once met their needs simply needs to be refined to continue to serve the customer's purpose. Another area that necessitates software modification involves debugging. Of course, software bugs are not desirable, but they often accompany applications. Even when solutions for bugs have been implemented, programmers unwittingly introduce new bugs.

Method

In order to compile a substantial document of the subject, Team B visited multiple websites to gain insight into the different possibilities of software maintenance procedures and processes. These resources provided a wealth of information, which led Team B to the following results

Results

Background

In order to control how an application is modified, a good software change control program is a necessity. It is important to control software changes within an organization because maintenance costs incur that largest portion of a software budget. Just in 2000, maintenance costs absorbed more than 90 percent of the software finances (Erlikh, as quoted from Koskinen, 2003).

The structured approach is the best way to control software maintenance. The product's design documentation is the best place to start. The design documentation provides the necessary background to better understand how to incorporate any modification. Programmers with a better grip of the application from its design standpoint help with the next step. That is, how the modification fits into the overall program. This involves establishing a plan to implement the change with the overall design in mind. The last portion of the structured approach involves review of the modification design, actual coding of the change, and testing (Stacey, 1995).

Software Maintenance Plan

This document can serve as a template for a software management and change control plan and to serve as a reference for management and developers in an organization. The objectives of the Software Management and Change Control Plan are defined as follows:

1. To ensure that no unauthorized requirement, design, or code change is made after a final version has been established.
2. To ensure that no version is changed by more than one engineer at any time.
3. To ensure that the impact of any change is evaluated, understood and managed.

Table 1

Change Management Roles and Task

<i>Roles</i>	<i>Task</i>
Manager	The individual responsible for managing the Change Management activities of the organization
Change Control Coordinator	A group or individual responsible for reviewing change requests for approval or disapproval.
Customer	Person or Business Unit representative requesting the change.
Quality Assurance Team	A group or individual who oversees the inspection of changes.
Developer(s)	Programmers responsible for code changes

The Manager is responsible for defining and managing the change process and the tools that support the process protecting the integrity of important versions. Management should provide each member with a copy of the organization Software Management and Change Control Policies. Once a project has been placed under change control, it must be monitored to avoid unauthorized modification.

Developers will use Microsoft Visual SourceSafe 6.0 for configuration management of all configuration items. Each Developer should have the client software, Microsoft Visual SourceSafe Explorer, installed on his/her computer. Visual SourceSafe and the Change Control database reside on the corporate server; therefore, they are backed up.

Data Restore will be done according to the recovery procedures defined by corporate IT Policy. Editing a file stored in Visual SourceSafe is done using a Check Out / Check In scheme. Checking Out a file places a copy of that file in the working directory on the user's machine. Other team members cannot access the file being edited while it is checked out to prevent accidental overwriting of changes. Other team members can see who has the file checked out by looking at the working directory field in the Visual SourceSafe Explorer file list. The working directory of the person who checked the file out will appear here. While a file is checked out from SourceSafe, all changes are made to the copy of the file that is stored locally. Changes are only committed when the file is checked back in to SourceSafe.

Checking In a file creates a new version of that file in the SourceSafe database. It is equivalent to committing the changes you have made to the item. The version that existed before the changes were made remains accessible through the file history, but the latest version is changed to the most recently checked in copy.

A version of a product that is considered complete when a particular set of requirements provides a checkpoint to which development can be backed out in the event that something unexpected occurs with the current version of the item. A version can be considered beta after it has successfully completed the Quality Assurance process as defined for the particular item. Successful completion of Quality Assurance occurs when the Quality Assurance Team approves the item.

Change Process:

1. The Customer submits a change request to the Manager.
2. Approval or Disapproval is sent to the Customer.
3. If request is approved, the team is notified of the request and they meet to discuss it.
4. The Manager documents the outcome of the meeting as part of the minutes.
5. The Manager notifies the client, if necessary, of the team's issues and/or concerns.
6. The Manager, based on the impact identified in the team's meeting, updates the plan and schedule as necessary.

A Change Request, including the following information:

1. Name and version of the configuration item with the problem
2. Originator's name, phone number, and organization
3. Date of request
4. Indication of urgency
5. Need for the change
6. Description of the change.

Beta products should be tested in development or production environments prior to code release for general use in daily production operations.

Discussion

In conclusion, the software maintenance plan provides an avenue of improvement and maintainability for the developer and customer. If this plan were not in place common software applications would be disjointed and unattainable. In other words, there would be anarchy. Most projects with more than one person must maintain some sense of balance and understanding. This is achieved through a plan. A plan establishes rules and guidelines requiring all users to act accordingly. This provides us with a well written, documented, and controlled software application that can be changed, upgraded, and improved with little interference from unknown procedures.

References

- Huy, P., Lewis, G., Liu, M. (2001, April 10). *Configuration Management Plan*. Retrieved June 5, 2005 from Carnegie Mellon University, School of Computer Science Web Site:
<http://dogbert.mse.cs.cmu.edu>
- Koskinen, J. (2003, Sep 12). *Software maintenance costs*. Retrieved Jun 1, 2005 from
<http://www.cs.jyu.fi/~koskinen/smcosts.htm>
- Stacey, D. (1995, Jul 28). *Software maintenance*. Retrieved Jun 1, 2005 from
<http://hebb.cis.uoguelph.ca/>