The Concepts Behind Software Maintenance and Regression Testing

Deborah J. Barndt

Illinois Institute of Technology

[dbarndt@hawk.iit.edu](mailto:dbarndt@hawk.iit.edu)

10-20-18

Abstract

[The abstract should be one paragraph of between 150 and 250 words. It is not indented. Section titles, such as the word Abstract above, are not considered headings so they don’t use bold heading format. Instead, use the Section Title style. This style automatically starts your section on a new page, so you don’t have to add page breaks. Note that all of the styles for this template are available on the Home tab of the ribbon, in the Styles gallery.]

Keywords: [Click here to add keywords.]

The Concepts Behind Software Maintenance and Regression Testing

Software changes continuously. Even after it is released, updates must be made to the system for it to adapt to the changing environment. Patches are then made available online for users to download. These patches could be because of new requirements that were needed, technology changes, new bugs were discovered, or new features being added after the software’s release. For example, after each mainline Linux kernel is released, the bug fixes are backported from a newer version and applied to the older version. Stable kernel updates are released on a required basis, which is usually once a week and new mainline kernels are released every 2-3 months (Linux Kernel Organization, Inc., 2018)! Thus, making software maintenance and regression testing a crucial part of software testing and the software development life cycle.

# Basic Concepts

## The Purpose of Software Maintenance

The main purpose of software maintenance is to modify and update software applications after it has been delivered to correct any faults and improve the performance (Bennett, Coleman & Co. Ltd., 2018). A common misconception about software maintenance is that it only involves fixing defects, which is not true. “Maintenance testing includes any type of testing of changes to an existing, operational system, whether the changes result from modifications, migration, or retirement of the software or system” (Black, Veenendaal, & Graham, 2017). Software maintenance is needed to repair defects that were missed in the released software, because defect removal and quality control processes are not perfect. It also must be carefully planned from the beginning of the life cycle phase and planning out your maintenance testing will help with creating a schedule for maintenance releases that may be needed in the future.

There are many factors that determine the need for maintenance such as correct faults, improving the design, implementing enhancements, interfacing with other systems, making the system adaptable, migrating legacy software, retiring software, or being able to provide continuity of service. In the end, software systems need to keep running, and much of our daily life is run by a computer. There could be catastrophic consequences if there is a system failure such as financial issues or serious disruptions. “Maintenance activities aimed at keeping a system operational include bug-fixing, recovering from failure, and accommodating changes in the operating system and hardware” (Grubb & Takang, 2007). A company also wants to be able to keep a competitive edge against your competitor’s products would involve mandatory upgrades. Furthermore, users will use the system more if it is a better product, which will get the users to request enhancements in the systems functionality or requirements for better performance. The last factor that determines maintenance is being able to facilitate future maintenance work. Taking shortcuts during software production can be very costly and could hurt the company financially in the long run. It could work out better financially and commercially to begin changes geared towards future maintenance, which would involve things such as database restructuring or updating the documentation (Grubb & Takang, 2007).

## The Key Issues and Challenges with Software Maintenance

Many of the problems that are associated with software maintenance usually involve the inadequacy of the development process. In order to effectively maintain those systems, it is essential to have a good base in the relevant theory. “Software maintenance is a key discipline, because it is the means by which systems remain operational and cope efficiently in a world ever more reliant on software systems” (Grubb & Takang, 2007). As a software maintenance engineer, you need to know how long you are maintaining the system whether it be for five minutes or five years. Software maintenance provides distinct technical and management challenges for software engineers (OpenStax, 2018). For example, introducing a new system and having to postpone the date of completion of the new system. This means that the older system has to be supported and maintained past its life expectancy. “The motivation to get it right comes from an understanding the wider implications, which in turn comes from an understanding of the theories and complexities of the discipline of software maintenance” (Grubb & Takang, 2007).

The software maintenance engineer needs to have a wide range of skills besides just computer programming, but they also need comprehension skills and analytical powers. Some of the range of skills that a software engineer needs are a limited understanding of where to make a change or correction in the software that the engineer didn’t develop. “Research indicates that some 40% to 60% of the maintenance effort is devoted to understanding the software to be modified” (OpenStax, 2018). Testing is another significant skill of a software engineer, and the cost of repeating full testing can be significant in terms of time and money to a company. Being able to coordinate testing when different members of the maintenance team are working on different problems can be a challenge when it comes to maintenance. The last skill that a software engineer will need is being able to possess close knowledge of the software’s structure and content, then using that to perform the impact analysis by identifying all the systems and products that will be affected by the change request.

# Types of Software Changes

# For the maintenance team or engineer to achieve the goals intended for the system, a broad-spectrum of changes may be necessary to the software. These changes can be classified into four types: corrective, adaptive, perfective, and preventive. Each of these changes can be classified individually, they are also intertwined with one another which is shown in Figure 1. Here we will go into detail about each of the changes and why they are important.

## Corrective Change

A corrective change makes a reference to the modification that is initiated by defects in the software. A defect could be design errors, logic errors, and coding errors. Design errors occur when changes that are made to the software are incorrect, incomplete, miscommunicated, or misunderstood. Logic errors occur when there are invalid tests and conclusions, the implementation of the design specifications was incorrect, faulty logic flow, or incomplete testing data. Coding errors occur when the implementation of the detailed logic design is incorrect and there is an incorrect use of the source code logic, making defects caused by data processing errors and system performance errors (Grubb & Takang, 2007).

Each one of these errors can prevent the software from adhering to the agreed specification. If there is a system failure due to an error, actions are then taken to try to restore the software system back to its normal operation. “Under pressure from management, maintenance personnel sometimes resort to emergency fixes known as ‘patching’. The ad hoc nature of this approach gives rise to a range of problems that include increased program complexity and unforeseen ripple effects” (Grubb & Takang, 2007). Program complexity usually comes from the degeneration of the program structure and thus makes the program more difficult to understand.

## Adaptive Change

An adaptive change refers to a change that is driven by the need to accommodate modifications in the environment of the software system. The term environment means the completeness of all the conditions and influences that are needed that act from the outside on the system. If there is a change to part or the entire environment it will cause a compatible modification to the software. “Adaptive maintenance includes any work initiated as a consequence of moving the software to a different hardware or software platform – compiler, operating system or new processor” (Grubb & Takang, 2007). A program can be changed because of new compiler that performs additional optimizations in order to create smaller and faster code. A program can also be modified to be able to take full advantage of the additional facilities that is provided by the new version of the system.

## Perfective Change

A perfective change refers to changes that are responsible for expanding the existing requirements of a system. For a software system to be successful, it is subjected to a succession of changes that result in increasing requirements. The increase in requirements is based on whether the software becomes useful, and the users experiment with the software beyond what it was developed for. An expansion of those requirements would then take on the form of enhancements of the existing system functionality. Programs can then also be catered to the users if some of the features are rejected or become redundant by being either removed or modified.

## Preventive Change

A preventive change refers to a change done to a software system to address a problem of a deteriorating structure. This type of change is used to prevent malfunctions or to improve the maintainability of the software itself and is usually initiated with the intention of making the program easier to understand. However, a preventive change does not usually lead to an increase in the original functionality. Also if a thorough impact analysis is not carried out in time before the change is made, unforeseen ripple effects could occur that could affect other sections of the program and be unpredictable as to the results which could lead to a distortion of the system (Grubb & Takang, 2007). “Program slicing techniques and concepts such as modularization and information hiding can be used to address these problems” (Grubb & Takang, 2007).

# Software Maintenance Processes

# [The first two heading levels get their own paragraph, as shown here. Headings 3, 4, and 5 are run-in headings used at the beginning of the paragraph.]

# [Heading 1]

[The first two heading levels get their own paragraph, as shown here. Headings 3, 4, and 5 are run-in headings used at the beginning of the paragraph.]

## [Heading 2]1

[To add a table of contents (TOC), apply the appropriate heading style to just the heading text at the start of a paragraph and it will show up in your TOC. To do this, select the text for your heading. Then, on the Home tab, in the Styles gallery, click the style you need.]

### [Heading 3].

[Include a period at the end of a run-in heading. Note that you can include consecutive paragraphs with their own headings, where appropriate.]

#### [Heading 4].

[When using headings, don’t skip levels. If you need a heading 3, 4, or 5 with no text following it before the next heading, just add a period at the end of the heading and then start a new paragraph for the subheading and its text.] (Last Name, Year)

##### [Heading 5].

[Like all sections of your paper, references start on their own page. The references page that follows is created using the Citations & Bibliography feature, available on the References tab. This feature includes a style option that formats your references for APA 6th Edition. You can also use this feature to add in-text citations that are linked to your source, such as those shown at the end of this paragraph and the preceding paragraph. To customize a citation, right-click it and then click Edit Citation.] (Last Name, Year)

References

Last Name, F. M. (Year). Article Title. *Journal Title*, Pages From - To.

Last Name, F. M. (Year). *Book Title.* City Name: Publisher Name.

Footnotes

1[Add footnotes, if any, on their own page following references. For APA formatting requirements, it’s easy to just type your own footnote references and notes. To format a footnote reference, select the number and then, on the Home tab, in the Styles gallery, click Footnote Reference. The body of a footnote, such as this example, uses the Normal text style. (Note: If you delete this sample footnote, don’t forget to delete its in-text reference as well. That’s at the end of the sample Heading 2 paragraph on the first page of body content in this template.)]

Potential Relationships of Software Changes

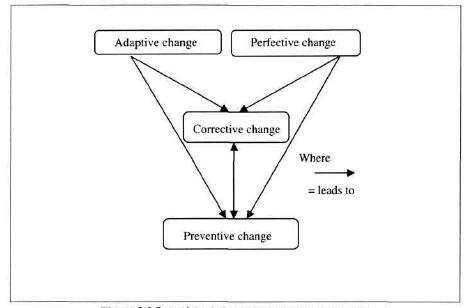


Figure 1. shows the potential relationships between each of the different types of software changes.

Tables

Table 1

[Table Title]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Head | Column Head | Column Head | Column Head | Column Head |
| Row Head | 123 | 123 | 123 | 123 |
| Row Head | 456 | 456 | 456 | 456 |
| Row Head | 789 | 789 | 789 | 789 |
| Row Head | 123 | 123 | 123 | 123 |
| Row Head | 456 | 456 | 456 | 456 |
| Row Head | 789 | 789 | 789 | 789 |

Note: [Place all tables for your paper in a tables section, following references (and, if applicable, footnotes). Start a new page for each table, include a table number and table title for each, as shown on this page. All explanatory text appears in a table note that follows the table, such as this one. Use the Table/Figure style, available on the Home tab, in the Styles gallery, to get the spacing between table and note. Tables in APA format can use single or 1.5 line spacing. Include a heading for every row and column, even if the content seems obvious. A default table style has been setup for this template that fits APA guidelines. To insert a table, on the Insert tab, click Table.]

Figures title:

Figure 1. [Include all figures in their own section, following references (and footnotes and tables, if applicable). Include a numbered caption for each figure. Use the Table/Figure style for easy spacing between figure and caption.]

For more information about all elements of APA formatting, please consult the APA Style Manual, 6th Edition.