CISC 615 Software Testing and Maintenance

Final Project

Jingda Guo

Feng Mi

Weiguo Lin

Shengling Qiu

Weida Zhu

Contents

[1 Proposed Maintenance Process 1](#_Toc437889889)

[1.1 Help Desk Model 1](#_Toc437889890)

[1.2 Key Source of Corrective Maintenance 2](#_Toc437889891)

[1.2.1 How We Measure the Priority 2](#_Toc437889892)

[1.2.2 The Ways We Collect and Assigns the Priority 2](#_Toc437889893)

[1.2.3 The Process of Collecting and Prioritizing Service Requests 2](#_Toc437889894)

[1.2.4 Details in the Collection and Prioritization 3](#_Toc437889895)

[1.3 Perfective Maintenance 3](#_Toc437889896)

[1.4 Preventive Maintenance 4](#_Toc437889897)

[1.4.1 Documentation Updating 4](#_Toc437889898)

[1.4.2 Code Optimization 4](#_Toc437889899)

[1.4.3 Code Restructuring 5](#_Toc437889900)

[2 Maintenance Team Structure and Methodology to Follow 5](#_Toc437889901)

[3 Special Considerations during Corrective Activities Initialization 6](#_Toc437889902)

[4 Connections between Maintenance and Development Team 7](#_Toc437889903)

[5 How Should the Code Base Be Managed? 8](#_Toc437889904)

[6 Testing in Maintenance Process 9](#_Toc437889905)

# 1 Proposed Maintenance Process

## 1.1 Help Desk Model

A Help Desk model needs to be employed to the application by People Resources. A Help Desk is a resource designed for customers or IT users to contact when they are having problems with their IT services. It enables companies to troubleshoot problems or provide guidance about their products such as computers, electronic equipment, or software. The application is developed by People Resources, with the target customers of small to medium sized organizations, for all facets of human resource processing including recruitment, benefits, payroll, and managing the employee life cycle from pre-hire to termination. Those organizations lack professional experience in such Human Resources software, and it is common for them to run into problems that they are not familiar with. Besides, waiting for problems to show up before setting up a Help Desk could only cost more in staff turnover, work disruption and loss of business. Furthermore, no plan of the processes and service offerings of a Service Desk is implied for this application. Thus, a Help Desk is appropriate to be adopted here.

A Help Desk institutes a multi-tiered trouble shooting approach by having personnel with extensive technical knowledge available. For most Help Desks, the service process begins with the customer call, which includes a phone call, Web submit, email, walk-in, FAX, or after-hour page. Since the application is developed for organizations across the United States and it is web-based, People Resources would have a lower chance to receive walk-in from customers. Instead, phone calls, Web submit, live chats, and emails are more likely to be accepted by customers. Experienced and trained employees should be staffed in the Help Desk to deal with the call flow. In this way, these employees can help resolve some of the calls to not only avoid unnecessary and costly escalations to an upper level but save consulting time for customers. The strategy of call flow that can be adopted by People Resources is the Specialist Model, which is the filtering of calls. Since most of the customers’ calls, in the real world, are about their confusion and forgetting their password, it would save much time if calls are sorted before the second-level Help Desk specialists set about figuring out the questions and providing feedback to customers. In this model, some of the calls can be solved by the first-level specialists. After these fundamental questions are sorted out, the rest calls are about maintenance. Then, the first-level specialists dispatch those rest calls to the more experienced and trained second-level specialists. This process helps reduce service levels and thus decreases the cost. People Resources can also migrate to more levels over time as the business grows and their needs evolve.

## 1.2 Key Source of Corrective Maintenance

### 1.2.1 How We Measure the Priority

Priority is used to establish timescales and effort to respond to and resolve an issue (incident or service request). Priority is derived from an impact and urgency Priority Matrix.

Impact: Measures the effect of an Incident/Service Request (ex: Number of Customers affected/influenced by the Incident).

Urgency: Measures how long it will be until the Incident has a significant impact on the business (ex: If the e-mail server goes down, all customers are immediately affected).

### 1.2.2 The Ways We Collect and Assigns the Priority

The initial prioritization is set by the Incident/Request Creator. A client does not have the ability to set the prioritization of their own ticket, nor do we expect our clients to know all the definitions and attributes required to objectively determine Priority levels. After working with the client to understand the situation, the initial assignment of Priority is completed by either the Creator or the Owner (depending on how the client contacts us).

### 1.2.3 The Process of Collecting and Prioritizing Service Requests

The creator classifies the ticket through the service category options, as well as setting the priority. The creator will have the option to nominate the ticket as a Major Incident if they believe this is a Priority 1 incident. Only Major Incident Owners can classify tickets as Priority 1 (or Major Incidents). A Major Incident represents the most significant type of service disruption that merits specific communications, coordination and handling protocol to ensure focus on fast restoration of service.

### 1.2.4 Details in the Collection and Prioritization

* Click the Priority drop-down magnifier to reveal the Priority Matrix (determined by invoked SLA), and then click a priority number.
* The priority is displayed in the Priority alert bar of the Quick Info Tile. The Priority bar displays the Incident or Service Request priority.
* After a priority is selected, respond by and resolve by target times are calculated based on the defined priority target times in the invoked SLA, and are then displayed in the SLA section of the Quick Info Tile.

## 1.3 Perfective Maintenance

Perfective maintenance involves making enhancements to improve processing performance, interface usability, or to add desired, but not necessarily required, system features.

So the maintenance team should have a group of people collect the user feedback and the operating information of every function of the app like recruiting, employee, data, Job function/data, Employee benefits, Payroll of the app separately. When the customer complain is too much and when the there are some new ideas of the enhancement may better cater the market we can consider making some perfective maintenance.

Aiming at different functions of the app like recruiting, employee, data, Job function/ data, Employee benefits, Payroll, we should have different evaluation specification to determine whether they need make some enhancements and focusing on the detail which can get better customer experience.

For example, the convenience and quicker information updating ratio in the database would bring good customer experience always taking these two factors into account when make the perfective maintenance.

## 1.4 Preventive Maintenance

After other parts of maintenance process have be done, maintenance teams will go through the perfective maintenance process. These types of maintenance performed for preventing problems before they occur by improving program understandability and increasing software maintainability. More specifically, three aspects will comprises in preventive maintenance process:

### 1.4.1 Documentation Updating

The previous process of maintenance make many change in the software for defect repairs and add new functions and features. Maintenance team need to modify the document to make it corresponding to the final product. How to change the software to a web-based application and include IOS and Android functionality, or change the data to cloud based will all be particularly detailed in document to make further maintenance in the future get a smoothly start in their project. These process will take the whole team to finish in one week.

### 1.4.2 Code Optimization

Code optimization is also a significant part of preventive maintenance. Whether the finished code has a good structure and under a good programming skill will be prior taken into account. These are more important for the old part of the code, they may wrote at old structure or technique, cause the less memory and storage capacity and may easily hurt the original structure. Resolving these problem are the main part of the code optimization. The QA team will spend 4 weeks to optimize the code, make it clear to be understood and easier for any subsequence changes.

### 1.4.3 Code Restructuring

After the code optimization, the maintenance team will continue to find any opportunities to reduce the complexity of source code and transforming. These process are used to make other engineers may deal with this product in future easier to understand the code. The main work of these process are make the structure of the added functional code consistent to the old part of the code. Clarity and conciseness are the goal of code restructuring. These process will be finish in two weeks.

# 2 Maintenance Team Structure and Methodology to Follow

We can structure the maintenance in this way: one manager to coordinate the whole maintenance team with original development team. And we can also organize five sub-maintenance teams aiming at the five functions. Each team should have one sub manager.

We need to notice that a good structure of the maintenance team would enable the company to provide beneficial results for its customers. How to structure the maintenance team depends on the type of work that People Resources is performing: planned and scheduled work. Thus, the maintenance team should focus on the majority of the work, 70%-80% of the work, in the Proactive column. It is important to identify the amount of weekly reactive work that the company has and assign portions of the maintenance team to do nothing but reactive work. This effort will allow the remainder of the maintenance team to concentrate on planned and scheduled work which should consist of preventive maintenance, corrective and normal repair requests. The amount of reactive and planned and scheduled work should be alerted so that the company can adjust the structure of the maintenance team to compensate for the changes in the customers’ needs.

The maintenance team should follow the methodology of Kanban, which would match inventory with demand and achieve higher levels of quality and throughput because of the just-in-time process. The team can use sticky notes on a whiteboard to create a picture of very transparent process of the work. The highly visual nature of Kanban enables the maintenance team to communicate more easily and efficiently on what work needed to be done and when. In the Kanban methodology, the team can be more flexible with re-prioritizing work whenever it is necessary and change can happen at any time as well.

# 3 Special Considerations during Corrective Activities Initialization

There are some special considerations should be made, they are as followed. Firstly, it is long term view concern. After the corrective activities are deployed, there is a problem that we must consider is how this corrective process will affect the system. Instead of just a correction to the system, is it a better decision to make an improvement to the malfunction part. For example, some users complain about the information retrieval is too slow, maybe there is something we can do to improve the source code to improve the software efficiency, or we can look at the algorithm that our code build on to solve the problem from the very basic part, so in this way can we preserve the value of the system better.

Secondly, it is maintenance testing concern. Right after a corrective process be implemented, we need to make sure it has no negative influence on the whole system, or have no potential possibility to jeopardize the whole system, this consideration have something to do with the testing process, the process which happens after the corrective patch be made, and before it is be deployed. Such concerns can be various. When a update is needed specifically in the data input as well as output interface, we need to make sure after we correct the interface malfunction, the data link between data base and end system still functional.

Third, it is time arrangement concern. When the corrective process be carried out, we must schedule our scrum team and estimate time that the update can be finished and tested. Because some emergency problem urge the maintenance work has to be a quick fix. It is not hard to imagine the situation that there is sudden crush down in the system that cause the all the users can’t login the system, as an people resource application with so many people visit and use every single day, we can see how devastating the problem is. So there is no doubt that the corrective process schedule must be considered.

Forth, it is security concern. Consider the special properties of our application, which is a people resource related application, that so many user data are conserved in our database, some of them could be rather private and need to be information secured. When we encounter a situation like this, a problem detected in the employee data system, which cause the system can’t recognize specific individual and shut out this user, when we make the corrective update to let the user back in our system, we need to make sure the nothing happens to undermine the confidentiality of the other employee’s data.

Fifth, it is maintenance team work-load concern. Since we have five scum team, but problems happens randomly, won’t falls in every section equally. Which means maybe in a situation there are five problems detected, all of them are related to the recruiting section of the system. As we can see here an arrangement to distribute the assignment properly is needed, because if we just simply throw all the work to the recruiting scrum team, this scrum team will just be overwhelmed by the huge work load not to mention the efficiency.

# 4 Connections between Maintenance and Development Team

The development team should be maintained as long as they are doing the developing since in agile, maintenance is kind of a regression process, as the development goes on the maintenance follows to ensure the system developed efficiently and detect and correct even evolve during the development process. In order to move fast, to make incremental changes and refactoring safe, the team needs a better safety net, by automating unit and functional tests and acceptance tests. It can take a long time to put in test scaffolding and tools and write a good set of automated tests. But even a simple test framework and a small set of core fat tests can pay back quickly in maintenance, because a lot changes (and bugs) tend to be concentrated in the same parts of the code-the same features, framework code and APIs get changed over and over again, and will need to be tested over and over again. It can take a long time to put in test scaffolding and tools and write a good set of automated tests. But even a simple test framework and a small set of core fat tests can pay back quickly in maintenance, because a lot changes (and bugs) tend to be concentrated in the same parts of the code-the same features, framework code and APIs get changed over and over again, and will need to be tested over and over again. To get this continuous testing to work, our team need a continuous integration environment. Our team need to understand, automate and streamline the build and get the CI server up and run and wire in tests and static analysis checks and report them.

5 How Should the Code Base Be Managed?

* Noticing that People Resources will have regular regulatory changes for example tax changes or personnel transfer and so on.
* In some cases there are emergent changes which will need the maintenance team make response in a quite short period.
* Making sure the responsibility of each team. The development team developed the software while QA team test the program and guarantee the quality of the app before it released.

However, our role is the maintenance team which is established after the production is released. So after the release of the app, if there are any changes need to be implemented we should do it ourselves at the first time. But remember that after we have made the changes we should tell the development team and test team because the changes may impact the next generation of the App and the development team and the test team could take the whole scenario into account by the virtue of the documentation we provide in this step.

* We can establish three sub-team, one is responsible for effective communicating with other groups, the second is responsible for handling the emergent issues and the third is responsible for handling issues for regular code change and we can absorb some of the original development team members into emergent team since the people who developed the product are more familiar with the code and they can address the critical and emergent issues more effectively.
* If the maintenance team decide to change the code of the application they must document the change very carefully in detail and let the development team and testing team know as soon as possible. When they begin to implement the code change they need have a quick meeting and report with other teams at least twice a week to report the progress.
* When there are no works for editing the code the team should assign people focus on collecting information and user feedback of the function: recruiting, employee, data, Job function/ data, Employee benefits, Payroll of the app separately in order to address the code changing issues more effectively when there are needs.
* We must take into account that if the changes we make would affect other parts of the app and evaluate the impacts before taking into action. If it is necessary communication with specialists and other teams is necessary.

# 6 Testing in Maintenance Process

After repair the bugs and add new functions, maintenance teams need to test both functional and non-functional aspects of the software. Unit testing, system testing and acceptance testing will be included. Whether the code runs well, the added function accord with the previous requirements and the interface between the database and software or Browser and software works satisfied. The goal of the testing in maintenance process should guarantee the finished product meet the requirements of maintenance and deliver the eligible software to public and customers. After the testing process finished, the software promoted to “live” and deliver to public to accept the feedbacks and comments from market.