# *Maintenance Project (420-E71-HR)*

# *Task 1 – Project Kick-off*

Date assigned: Monday, January 22 2018

Date due: **Monday, January 22, 2018, 12:00pm (noon)**

**Learning Objectives**

Upon successful completion of this task, the student will be able to:

* Review/Understand what Maintenance Projects are about
* Review/Understand refactoring

To do:

**Task Set-Up**

1. Save this document with the name **YourUserName\_E71\_A01\_Project\_Kickoff.docx** in the 420E71 folder of your H drive.

**Part A – Project Maintenance**

**TO BE DONE INDIVIDUALLY**

1. Go to the web page <http://www.slideshare.net/nashjain/agile-maintenance> and answer the following questions.
   1. According to this author what is maintenance defined as?

The process of modifying a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment.

* 1. The proliferation of mobile technology is likely to result in one type of maintenance occurring more often. Which type of maintenance is likely to occur more often and why? (This is not part of the slides…you need to think about it).

You’ll need to add the requirement for many websites to have them be mobile friendly, or in some cases come in the form of an actual app. This is a growing requirement and something part of maintenance for existing applications.

* 1. Given what you have been told about the systems we will be working on this term what type of maintenance are we performing on CSAdmin?

I think we’ll mostly be doing corrective and perfective maintenance. Improving exists parts of the system to make it faster, as well as creating new functionality for the system.

* 1. What is software configuration management? What best practices are involved in Configuration Management?

The task of tracking and controlling changes in the software.

* + - * Configuration Identification
      * Configuration control
      * Configuration status accounting
      * Configuration auditing
      * Build management
      * Process management
      * Environmental management
      * Teamwork
      * Defect tracking
  1. What is the cycle for completing an iterative enhancement?
     + - Analysis
       - User acceptance and Unit tests
       - Redesign and implementation
       - Regression tests
       - Release
  2. The author lists 8 reasons why maintenance is expensive. Of these reasons which do you think are the top three and why?
     1. Planning
        + People never plan for maintenance and it ends up costing more money
     2. Program age and structure
        + You need to hire people who have the technical skills required to work on the project, which can often be a niche market for older applications
     3. Staff skills
        + If you can’t find people who have the technical knowledge already or can’t afford people who have the knowledge, you need to train a team to work on the maintenance.
  3. In my opinion, the author’s two cents make a lot of sense. What are his two cents and briefly discuss whether or not you believe each of them.
     1. Source code is king
        + Well commented source code provides all the documentation you need, and source code will always reflect the current state of the project. You shouldn’t look at external documents to determine what’s in the source.
     2. Untrustworthy documentation
        + As soon as you finish writing documentation, it’s out of date. Documentation can’t keep up with constantly changing requirements for an application.
     3. The bug-tracking database stores knowledge
        + Knowing the flaws in the system tells you where you need to focus on the system, which is important to maintaining something properly.
     4. Reproduction is essential to obtaining a solution

1. Go to the web page <http://www.slideshare.net/rajeevsharan/software-maintenance-9841842> and answer the following questions.
   1. What are the six purposes of software maintenance?
      1. Equipment reliability
      2. Least operating costs
      3. Defect elimination
      4. Maximum production
      5. Risk reduction
      6. Failure avoidance
   2. What are the four types of software maintenance? Provide a description of each type in your own words. Provide an example of each type of maintenance as it relates to something non-software and something software.
      1. Corrective maintenance
         * Maintenance performed to correct faults in hardware or software
      2. Adaptive maintenance
         * Maintenance performed to make a computer program usable in a changed environment
      3. Preventative maintenance
         * Maintenance performed for the purpose of preventing problems before they occur
      4. Perfective maintenance
         * Maintenance performed to improve the performance, maintainability or other attributes of a computer program
   3. The author talk about six software maintenance processes. List them (briefly) and order them in terms of difficulty and importance (the order does not have to be the same). Briefly, justify why you chose that order.
      1. The implementation process
      2. The problem and modification analysis process
      3. Implementation of the modification
      4. Acceptance of the modification
      5. The migration process
      6. Retirement of a piece of software
   4. According to the author, what are the causes of software breakdowns?
      * + Basic conditions neglected
        + Inadequate skills
        + Operating skills not followed
        + Deterioration unchecked
        + Inherent weak design.
   5. What is Optimal Maintenance? What parameters are often considered? Why is this important?

It’s the discipline which is concerned with maintaining a system in a manner that maximizes profit or minimizes costs.

* + 1. Cost function depending on the reliability and maintainability characteristics of the system determine the parameters of interest to minimize.
       - Parameters often considered are:
         1. The cost of failure
         2. The cost per unit of “downtime”
         3. The cost per time unit of corrective maintenance
         4. The cost per time unit of preventative maintenance
         5. The cost of repairable system replacement

1. Go to the web page <http://www.slideshare.net/deepikashanti/12-software-maintenance> and answer the following questions.
   1. What are some of the sources from which maintenance requests can come?
      1. New business or market conditions which cause changes in production requirements or business rules
      2. New customer needs that demand modification of data, functionality or services delivered by the system
      3. Reorganization and/or business downsizing that changes priorities and team structure
      4. Budgetary or scheduling constraints that cause a redefinition of the system
      5. MOST CHANGES ARE JUSTIFIED
   2. This slide set lists a number of reasons why maintenance is so costly (consuming between ½ and 2/3 of the software development budget). He lists 11 different reasons over 3 slides. Select the two reasons that you think are most important, in general, and discuss why you consider them the most important.
      1. Age of the system
         * You need to find people who know the technology used to build the system. If you need to maintain an old COBOL system, you need to find someone who knows COBOL. And finding someone is hard and therefore very expensive.
      2. Quality of program documentation
         * Documentation will always be out of date, so it is going to be an unreliable source of information for you to consult when maintaining a system – and source code is usually scarcely commented.
   3. What differences between new development and maintenance are discussed?
      1. Constraints of an existing system
         * Changes must conform or be compatible with and exististing architecture, design and code constraints.
      2. Shorter time frames
         * Development spans 6 months upwards
         * Maintenance spans hours or days up to 6 months
      3. Available test data
         * Development creates all test data from scratch.
         * Maintenance uses existing test data with regression testing, creating new data for the changes.
2. Go to the web page <http://www.slideshare.net/xebiaindia/agile-maintenance-10> and answer the following questions.
   1. Briefly explain in your own words the issues with the traditional approach to maintenance.

The first glaring issue is that you’re waiting for bugs to come in to a support team. Instead of proactively maintaining the system and having teams of testers checking for bugs, you’re waiting for them to come in. You also have a lot of unnecessary documentation and you need to run your tests manually constantly.

* 1. The author discusses a one team multiple project approach to maintenance. He discusses a number of advantages to this approach. What are they? List at least two disadvantages of this approach and discuss how important they are relative to the advantages.
     1. You have your attention divided across multiple projects where you don’t know either system quite as well.
     2. One project may end up getting neglected if people simply care about the other one more.

These disadvantages do not warrant not having one team working on multiple projects, since that will save both a lot of time and a lot of money, while keeping developers more stimulated.

* 1. The author suggests that Agile Maintenance should use a “Type C Scrum”. Explain what that is and why it is a good idea for maintenance (you may need to do other research for this).

This is a good idea for maintenance because there’s fixes to be done across all parts of the system. If you don’t worry about which sprints you’re really working in, and all of them kind of overlap, then you end up with getting to all of the issues in the system a little bit quicker than if you do it in a type A or B scrum.

**Part B – Refactoring**

**TO BE DONE INDIVIDUALLY**

1. Use the web page <http://www.slideshare.net/vorleak.chy/introduction-to-refactoring> to answer the following questions:
   1. What is refactoring?

The process of changing a software system in such a way that it does not alter the external behavior of the code, yet improves its external structure.

* 1. Why is refactoring needed?
     + Helps us deliver more business value faster
     + Improve code structure and design
     + Keep development at speed
     + Make the software easier to understand
     + To help us find bugs.
  2. When should you refactor?
     + Add new features
     + Fix bugs
     + During code review
     + Only refactor when refactoring. Do not add features while refactoring.
  3. The author suggests two times you shouldn’t refactor. What are they? Explain what he means in your own words.
     + Sometimes you should throw things out and start again
       1. The module was poorly designed and not simply poorly written. It should be re-built with a better design.
     + Sometimes you are too close to a deadline
       1. Even if something really should be refactored, sometimes you just don’t have the time to do it before the deadline and it’ll need to wait until the next iteration.
  4. When refactoring you look for “code smells”. What are some of the common “code smells” and provide a brief explanation as to why they are “smells”
     + Duplicate Code
       1. Need to maintain the code in 2 places now, which is bad
     + Feature envy
       1. You have a method pertaining to external classes more than the one that it is in, and is likely in the wrong place.
     + Comments
       1. If there’s too many comments, it’s because the code isn’t clear enough
     + Long methods
       1. Long methods will decrease readability
     + Long parameter lists
       1. Likely can be an object instead of a list of parameters if the list gets too big or repeats to other methods.
     + Switch statements
       1. Often repeated and quite bulky

1. Use the website <http://jamesshore.com/Agile-Book/refactoring.html> to answer the following questions.
   1. In a couple of sentences describe, in your own words, reflective design.

Reflective design is taking a look at your code and finding anything that seems slightly of (“code smells”), taking a closer look at it, and considering it for refactoring.

* 1. How does the author say refactoring should be approached with respect to reflective design?

Reflective design is looking at what needs to be changed, and refactoring is actually going and implementing that change.

* 1. What is needed for effective refactoring?

You need to be able to effectively identify issues with your code so that you can go and write up the issues and fix them when refactoring.

Mark Breakdown:

|  |  |  |
| --- | --- | --- |
|  | **Marks** |  |
| **Part A – Project Maintenance** |  |  |
| 1 Responses a-g (2pts x 7) | 14 |  |
| 2 Responses a-e (2pts x 5) | 10 |  |
| 3 Responses a-c (2pts x 3) | 15 |  |
| 4 Responses a-c (2pts x 3) | 15 |  |
| **Part B – Refactoring** |  |  |
| 1 Responses a-g (2pts x 5) | 10 |  |
| 2 Responses a-c (2pts x 3) | 15 |  |
| Assessment, English, properly handed in | 7 |  |
|  | 86 |  |

**To submit**

When you have completed the task:

* Save this file as username\_E71\_A01\_ProjectKickOff.docx and copy to the appropriate location in Moodle.