ENPM808X: Week 4:Writing Assignment
by Sumedh Reddy Koppula, UID: 117386066

5.1 How are software changes classified by their purpose? What is the most common purpose of the change?

Ans: Software changes are classified into four types depending on their purposes:

1. Perfective changes
2. Adaptive changes
3. Corrective changes
4. Protective changes

Perfective changes: (most common)

They introduce new functionality and increase the value of software. An example is the introduction of a credit card payment to the point-of-sale system that had only cash payment before; such a change extends the functionality of the program and increases its value.

Adaptive changes:

An example of such circumstances is a new version of the operating system or a new compiler used in the project, and so forth.

Corrective changes:

These correct software bugs and malfunctions which impact the users in often unexpected ways.

Protective changes:

These are invisible to the users, and they shield the software and its value in a proactive way. Examples of such changes are changes that improve the structure of software to make the future changes easier, and hence, they represent an investment that will pay off over time.

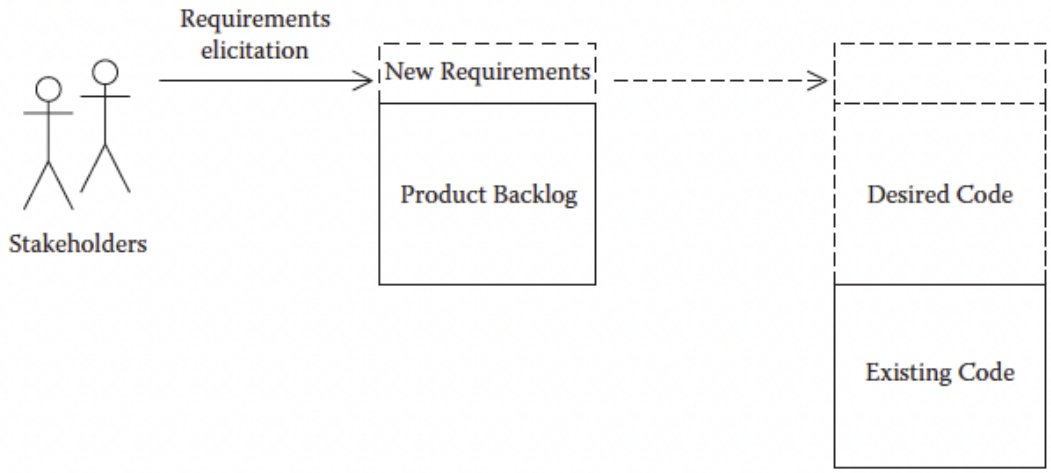
5.3 When is it permissible to do quick-fix changes?

Ans: The accept able circumstance for a quick fix during the evolution stage is in the situation of an emergency, where human life or a substantial value is at stake; thus, the fix has to be done quickly, and the speed outweighs every other consideration. Quick fix is a common strategy of change during the servicing stage, there is no reason for any gold plating, and the change is done as cheaply as possible. The patchwork left in the wake of quick fixes does not matter because the value of the software is low anyway, and any additional lowering of the software value is inconsequential.

5.5 What is a product backlog?

Ans: There is a whole set of requirements that programmers have to manage. The set of requirements is stored in a product backlog. The product backlog is also called a requirements

data base, and in other contexts it is called a project wish list because it lists desired future product properties and functions.



6.6 Describe a situation when a grep search fails. What would you do if this happened to you?

Ans: If the code does not have meaningful identifiers or comments, the grep search utility does not work. If there are no clean and understandable contracts, the dependency search does not work either. A decayed structure of software may complicate unit testing or code inspection and, as a result, the changes become increasingly difficult and risky. It may reach the point where further evolution is beyond the capabilities of the programming team, and this pushes the software into the **servicing stage.**