Book Questions
Chapters 9
Devayan Mandal

9.1 Explain why a software system that is used in a real-world environment must change or become progressively less useful.

Ans:

1. There are several real-world environment applications of software systems which require software updates on a continuous basis in order to address the ever changing needs of end users. A few important real-world scenarios which necessitate continuous software development include:

    a. Businesses which extensively implement software systems in the core requirements are corporations - water, paper, milling, chemicals, information technology, electricity, stock markets and trading organizations. A minor lag in software system updates would greatly affect these industries.
    b. Academic institutions.
    c. Law related and political environment.
    d. Sports and athletic enterprises.
    e. Agricultural, farming and natural resource management organizations.

2. Software systems implemented in any of the above industries or enterprises much continually ensure that specific business requirements are met in order to prevent the system from becoming obsolete.

3. The software system also need to compete with other systems which are released in the market on a continually basis so that it can provide greater if not the same end user benefits as the software competitors presently in the market.

9.4 Under what circumstances might an organization decide to scrap a system when the system assessment suggests that it is of high quality and high business value?

Ans: An organization may scrap a particular software system, even though it's current assessment suggests a high quality and high business yield for reasons which may include:

a. The market for the software system is short lived and does not offer steady income or profits over the long term.

b. The software system does not enjoy readily available support for software and/or hardware needs. Reporting of errors and technical difficulties is imminent and without a well-staffed support team the system is bound to become a failed business venture.

c. Also, even if the system boasts of a fully-equipped support team, a complete overhaul of hardware may require scrapping the present software system too.

d. Cost of maintenance of the current system is excessive.

e. The goals of the end-user(s) requirements changes and the changes are so expansive that the cost and time required to modify or improve on the present software system is much more than developing or purchasing a newer system.

9.5 What are the strategic options for legacy system evolution? When would you normally replace all or part of a system rather than continue maintenance of the software?

Ans: Strategic options for evolving a given legacy system depends greatly on the system under consideration. The following options may apply as a framework for approaching the process of evolution.

1. The maintenance of the present system ensues. Changes, modifications or additions within the maintenance aspect, such as hiring new talent who understand the revised needs of the end-user or purchasing new software or hardware tools which enhance maintenance can be implemented.

2. Similar to revising components of the maintenance aspect, revisions and improvements can be made in the software of the present system itself so that it can bring in more ease in the software functionality and maintenance.

3. A strategic option for continuing on the same system and maintenance scheme might entail the use of wrapper classes. A given system component may be wrapped with functionality which addresses the new requirements of the system and also syncs itself with the existing maintenance tools. The development team can proceed to wrap as many components within the system as it is deemed feasible to continue using the same system and maintenance specifications.

4. As discussed in the 9.4, if maintenance costs of the present system are too high, then it might be decided to scrap the system and bring in a new one.

9.8 Briefly describe the three main types of software maintenance. Why is it sometimes difficult to distinguish between them?

Ans: The three main types of software system maintenance are:

a. Perfective maintenance - this type of maintenance entails the addition of new functionality to the existing system.

b. Corrective maintenance - this type of maintenance entails addressing faults, bugs and errors which may show up during execution of the software system.

c. Adaptive maintenance - this type of maintenance entails making modifications to the software so that it adapts seamlessly with the ever changing real-world environment.

It may be difficult to distinguish between the three types of software maintenance because the functionality of each type may overlap with the other in many different situations. Also, the three types work very cohesively - almost as a single unit. For example, a scenario may arise wherein the software system observes a temporary crash due to a new software virus. The component of adaptive maintenance comes into play because it updates the database of newly identified viruses and relays that over to the maintenance team. The component of corrective maintenance comes into play because it addresses any errors or crashes in individual system components. Finally, the perfective maintenance piece ensures that modifications to the present maintenance protocol includes protection against the newly identified virus and hence strives for smooth operation of the system.