Plenary Talk - 5

A Practitioner's Approach to Software System Design

Debajyoti Mukhopadhyay

Web Intelligence & Distributed Computing Research Lab
Calcutta Business School, D.H. Road, Bishnupur 743503, India
debajyoti.mukhopadhyay@gmail.com

Abstract: How to design an efficient and cost effective software system is a million dollar question to every software engineer. The challenges lie in the fact that there is no such fixed rules and regulations, following which will produce an efficient software system. Practitioners have proposed many solutions to address this issue. In this discussion, we will try to touch base on some of the vital points which a practitioner is encouraged to adapt in order to lead a team of software engineers to produce cost effective efficient system.

Keywords*:* Software Development Life Cycle, SRS document, Cohesion, Coupling, , Re-usability.

1. Introduction

The job of a software engineer is to find best way of producing software product. We will try to find out what are the best practices in order to achieve that goal. It is important to understand what is meant by software product. Software products fall into two broad classes: A) Generic Products: These are stand-alone systems which are produced by a development organization and sold on the open market to any customer who is able to buy them. B) Bespoke (customized) Products: These are systems which are commissioned by a particular customer. The software is developed specially for that customer by some contractor.

The procedures of software development and the software engineering methods which may be used are the same for software products and bespoke systems. The most significant difference is that generic product specifications are produced internally by the marketing department of the product company. They reflect what they think will sell. They are usually flexible and non-prescriptive. By contrast, specifications for bespoke systems are often the basis for the contract between customer and contractor. They are usually defined in detail and changes have to be negotiated and carefully estimated in terms of cost.

2. Software Product Attributes

Like all engineering, software engineering is not just about producing products but involves producing products in a cost-effective way. Given unlimited resources, the majority of software problems can probably be solved. The challenge for software engineers is to produce high-quality software with a finite amount of resources and to a predicted schedule.

The attributes of a software product are the characteristics displayed by the product once it is installed and put into use. These are not the services provided by the product. Rather, they are concerned with the product's dynamic behavior and the use made of the product. Examples of these attributes are therefore maintainability, dependability, efficiency, usability, and so on.

Maintainability: It should be possible to evolve software to meet the changing needs of customers.

Dependability: Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure.

Efficiency: Software should not make wasteful use of system resources such as memory and processor cycles.

Usability: Software should have an appropriate user interface and adequate documentation.[1][2][3][4]

3. The Software Process

The software process is the set of activities and associated results which produce a software product. These activities are mostly carried out by software engineers. Computer Aided Software Engineering tools (CASE tools for short) may be used to help with some process activities.[5]

There are four fundamental process activities which are common to all software processes. The activities are:

1. Software Specification: The functionality of the software and constraints on its operation must be defined.
2. Software Development: The software to meet the specification must be produced.
3. Software Validation: The software must be validated to ensure that it does what the customer wants.
4. Software Evolution: The software must evolve to meet changing customer needs.

978-0-7695-4294-2/10 $26.00 © 2010 IEEE
DOI 10.1109/ISED.2010.13

There is no such thing as a "right" or a "wrong" software process. Different software processes decompose these activities in different ways. The timing of the activities varies as does results of each activity. Different organizations use different processes. However, some processes are more suitable than others for some types of application. If the wrong process is used, this will probably reduce the quality or the usefulness of the software product to be developed.

Detailed software process models are still the subject of research but it is now clear that there are a number of different general models or paradigms of Software Development Life Cycle:

1. Linear Cycle or The Waterfall Model: This takes the above activities and represents them as separate process phases such as requirements specification, software design, implementation, testing and so on. After each stage is defined it is "signed-off" and development goes on to the following stage.
2. Evolutionary Development: This approach interleaves the activities of specification, development and validation. An initial system is rapidly developed from very abstract specifications. This is then refined with customer input to produce a system which satisfies the customer's needs. The system may then be delivered. Alternatively, it may be re-implemented using a more structured approach to produce a more robust and maintainable system.
3. Formal Transformation: This approach is based on producing a formal mathematical system specification and transforming this specification, using mathematical methods, to a program. These transformations are "correctness-preserving." This means that one can be sure that the developed program meets its specification.
4. System Assembly from Reusable Components: This technique assumes that parts of the system already exist. The system development process focuses on integrating these parts rather than developing them from scratch.

The first two of these approaches, namely the waterfall approach and evolutionary development, are now widely used for practical systems development. Some systems have been built using correctness-preserving transformations but this is still an experimental process.

3.1 The Waterfall Model

The waterfall model or the linear cycle has been widely used in the design of systems that can be easily understood and which have well-defined workflows. It is a process that specifies a sequence of steps to be followed by designers. The linear cycle groups system development activities into a sequence of consecutive phases. A phase in the sequence can only commence once the previous phase has been completed. A report is produced at the end of each phase, describing what has been achieved and outlining the plan for the next phase. The report also includes any system descriptions, new or expanded user requirements, design decisions and problems encountered. This information is used during the next phase. Phase reports are also used to keep management informed of project progress, so that management can use the reports to change project direction and to allocate resources to the project.

Linear cycle phases are chosen to encourage top-down problem solving. Designers must first precisely define the problem to be solved and then use an ordered set of steps to reach a solution. The linear cycle gives the project direction and guidance about what should be done as the project proceeds. The system life cycle also assists management by producing reports on project status and keeping track of resource needs.

The top-down approach implies that both the problem to be solved and the solution are elaborated in more detail as the project proceeds. Thus the linear cycle begins with a broad definition of requirements and continues by developing the solution in greater detail. The first step in such detailed development is to determine the feasibility of providing a workable solution. If a solution is deemed feasible, then a detailed analysis of the system is made in order to produce both a description of the system and an expansion of user requirements of the new system. Analysis is followed by system design and implementation. These linear cycles phases are described below in detail.

It is also important to take precautions to ensure that all user requirements are satisfied at the completion of the project, that is, that the necessary quality assurance mechanisms are in the process. It includes validation of outputs at the different steps against user requirements. Validation takes place at the conclusion of the feasibility study, systems analysis and design phases and then finally following implementation. Verification takes place continuously through the process, with the outputs at one phase verified against the inputs from the previous phase. Finally, any designed software modules must be thoroughly tested during development.

3.2 Phases in Waterfall Model

3.2.1 Phase 1 – Problem Definition

Many people consider problem definition to be the most important phase. It defines the user requirements, or what the user expects the system to do, and thus sets the direction for the whole project. This phase also sets the project bounds, which define what parts of the system can be changed by the project what parts are to remain the same. The resources to be made available to the project are also specified in this phase. These three important factors: the project goal, project bounds and resource limits – are sometimes called the project's terms of reference. Because of their importance, they are set by the organization's management.

The output of this phase defines the user requirements. It consists of the project goal, its bounds and the terms of reference for the project. Furthermore, it may include any

restrictions on the project, such as the parts of the existing system which cannot be changed, as well as those which can.

Resource limitations are also often specified at this time to indicate the funds and personnel available for the project. This will include a rough idea of the resource requirements of each of the subsequent project phases. It will contain tentative start and completion dates for each phase and the number of persons expected to be involved in each phase.

3.2.2 Phase 2 – Feasibility Study

The feasibility study proposes one or more conceptual solutions to the problem set for the project. The conceptual solutions give an idea of what the new system will look like. They define what will be done on the computer and what will remain manual. They also indicate what input will be needed by the systems and what outputs will be produced. These solutions must be proven feasible and a preferred solution must be accepted.

Three things must be done to establish feasibility. First, it is necessary to check that the project is technically feasible. Does the organization have the technology and skills necessary to carry out the project and, if not, how should the required technology and skills be obtained? Second, operational feasibility must be established. To do this it is necessary to consult the system users to see if the proposed solution satisfies user objectives and can be fitted into current system operation. Third, project economic feasibility needs to be checked. The study must determine whether the project's goal can be achieved within the resource limits allocated to it. It must also determine whether it is worthwhile to proceed with the project at all, or whether the benefits obtained from the new system are not worth the costs (in which case, the project will be terminated).

Many beginners in systems analysis find the idea of a conceptual solution hard to understand. All that is needed at this stage is a very broad idea of the solution, enough to give potential users an estimate of whether it can work and how much it is likely to cost. It is usually a good idea to consider a number of possible solutions at this stage.

One output of a feasibility study is a preferred conceptual solution together with the expected system costs and benefits. It also includes a more detailed specification of what is needed of the new system. The plan in Phase 1 is now expanded to show the resource requirements for each phase. The phase output is validated against the user requirements.

3.2.3 Phase 3 – Systems Analysis

This phase is a detailed appraisal of the existing system and included finding out how the system works what it does. It also includes finding out in more detail what the system problems are and what users require of any new or changed system. Following this phase, analysts should be familiar with the detailed operation of the system and what is required of the new system. Analysts use many of the commonly used systems analysis techniques, such as data flow diagrams and data analysis. They also follow specified procedures in their search for information about the system. This calls for interviews with system users, questionnaires and other data-gathering methods. Analysts must spend considerable time examining components, such as the various forms used in the system, as well as the operation of existing computer systems.

This phase results in a detailed model of the system. The model describes the system functions, system data and system information flows. The phase report contains any revisions to project goals and cost-benefit estimates. It also contains a more detailed exposition of system problems than that given in Phase 1. This exposition usually includes a detailed specification of user requirements. Such requirements are used to set the objectives for the new system. Part of the management review of this phase is to reach an agreement on such objectives. Once agreement is reached, design can commence. The output is validated against requirements and the existing system. Validation in this case often incorporates the ideas of structured walkthroughs.

3.2.4 Phase 4 – System Design

This phase produces a design for the new system. There are many things to be done here. Designers must select the equipment needed to implement the system. They must specify new programs or changes to existing programs, as well as a new database or changes to existing database. Designers must also produce detailed procedures that describe how users will use the system.

System design usually proceeds in two steps: broad design (Phase 4A) and detailed design (Phase 4B).

3.2.4.1 Phase 4A – Broad Design

During this phase, the conceptual solutions proposed by the feasibility study are looked at in more detail. Major new functions are proposed and changes to existing functions defined. Important inputs and outputs are also defined at this point and performance requirements are specified. In addition, broad design outlines which part of the system is to be automated and which will remain manual. Thus, in our case, we may still consider such decisions as whether to place the actual ordering on the computer or to have manual ordering and only use the computer for dissemination of data. Alternative automation boundaries may therefore be considered and their costs evaluated in this phase.

At the conclusion of broad design we may know what we need in order to build the system. This may include the size of the computer and the software needed to put the system together. It will also state which software can be purchased off-the-shelf and which requires new programs to be developed. The design may also suggest whether the computer should be rented or purchased and whether

programs are to be developed using internal programmers or external contractors.

The broad design identifies the main architecture of the proposed system. This architecture is verified against the proposed system model and validated against user requirements.

3.2.4.2 Phase 4B (Detailed Design)

It is only when a broad solution is chosen that detailed design starts. During the detailed design phase, the database and program modules are designed and detailed user procedures are documented. The interaction between the system users and computers is also defined. These interfaces define exactly what the user will be expected to do to use the system.

The output of detailed design includes a proposed equipment configuration together with specifications for the database and computer programs. Detailed user procedures are also provided. These include any input forms and computer interaction between users and the computer. The user manual is also ready at the end of this phase. This output, particularly the proposed interactions between the computer and users, is validated against user requirements.

3.2.5 Phase 5 – System Construction

Like the design phase, this phase is also often broken up into two smaller phases: development and implementation. The individual system components are built during the development period. Programs are written and tested and user interfaces developed and tried by users. The database is initialized with data. During implementation, the components built during development are put into operational use. Usually this means that the new and old systems are run in parallel for some time. To complete the changeover, users must be trained in system operation and any existing procedures converted to the new system. One important part of construction is testing. It is necessary to test all modules to ensure they work without any errors once they are put into operation. Testing itself can be a process on its own. We first test individual modules, then we test their interfaces and see how they work together. At the end of this phase, users are provided with a working system.

4. Conclusions

We have discussed the basic steps involved in designing a cost effective and efficient software system. It is important to carry out each of the phases in the life cycle model. Meeting the users several times and collecting sample data and reports always help writing a good Software Requirements Specification document. While designing the software system, creating modules with high cohesion and low coupling is always desirable. Reusability is another important aspect for creating cost effective system. Special attention should be given to develop the software such that it is modular in nature and can be re-used as much as possible instead of in re-inventing the wheel thus saving time and cost.

References

[1] Pressman, Roger S., “Software Engineering: A Practitioner’s Approach”, Tata McGraw-Hill, Fifth Edition, 2001.

[2] Jalote, Pankaj, “An Integrated Approach to Software Engineering”, Narosa, Third Edition, 2005.

[3] Sommerville, Ian, “Software Engineering”, Pearson Education, Sixth Edition, 2002.

[4] Mall, Rajib, “Fundamentals of Software Engineering”, Prentice-Hall India, Second Edition, 2003.

[5] Lawrence, Shari, “Software Engineering: Theory and Practice”, Pfleeger; Pearson Education, Second Edition, 2001.

[6] Mukhopadhyay, Debajyoti and Dalezman, Blanka, “Designing Open Systems with CASE”, Information Systems Management Journal; Auerbach Publications, Boston, Massachusetts, USA; Vol.12 No. 1; Winter 1995; pp.26-34.