Software Design: Process and Quality Assessment

Dr. Qais Faryadi
Faculty of Science and Technology (FST)
Department of Computer Sciences
Universiti Sains Islam Malaysia (USIM)
Malaysia

Abstract

This assessment argues that one of the most vital functions of system engineering is the identification of the problem. It describes the functions that a system must perform to satisfy the needs of the target clients. Stating the problem also helps engineers to identify what functions their system requires. In system engineering, it is important to know what needs to be done to the system to arrive at the desirable outcome. Furthermore, we need a good theory that is grounded on a viable philosophy so that we can adopt appropriate approaches for software development. The applications of software range from the simple to the complex. In almost every sector of modern life, software has revolutionized everyday living, helping to solve problems and making our lives safer and more comfortable. This assessment also discusses interesting topics such as software process, procedure, system-engineering life cycle, system design and takes a serious look into software quality and measurement.

Keywords: Software, engineering, design, software development and quality measurement

Introduction

Software has become an important part of our lives. Software that is designed for household appliances has reduced the drudgery of many household chores. Not only has it enabled us to save time and effort, but it has also made modern living more comfortable and enjoyable. For example, consider the function of a toaster. It toasts our bread according to a pre-determined code and toasts our bread the way we want. At the click of a mouse, we can pay bills online. Therefore, good software must be able to fulfill a specific need or solve a certain problem. When developing software, one must consider quality rather than quantity. It is a challenging task, and requires a nimble mind and expertise on the part of the engineers.

Firstly, engineers have to look at the history of software successes and failures. What software is already available in the market? What do we hope to achieve in future with the new software? How is the development of quality software monitored? Who are the relevant personnel for software development? What is the demand from the targeted customers? The answers to these questions would provide a concrete grounding for the design of good software. We need software that can cater to the demands of the client. It should help provide solutions to problems faced by the end user or enhance the quality of work performance.

The Process

Problem Investigation

In order to find a solution to an identified problem, engineers should first carry out a thorough investigation and analysis of the problem. Dissecting the problem into small parts is very useful for better understanding so that relevant experts can be assigned to handle each section. Engineers need skills in programming languages, thinking, calculations, and drawings to develop effective, user-friendly software. Therefore, the computer becomes a tool for engineers to find solutions to problems.

Furthermore, engineers need to apply carefully articulated procedures to build the desired software. As an analogy, if you want to carry out any research, you should state your methodology. Similarly, in developing software, start by having a topic of enquiry.

Next, write your introduction before presenting the problem statement, literature review, methodology, results and finally conclusion and discussion. These are tools to help build good software with effective programming. An example of useful software is Microsoft Office Word. Instead of using a typewriter to type your letter, it is more efficient to use this software to write as you can delete, or cut and paste to re-arrange different parts of your letter.

Procedure

Engineers develop software using a combination of tools and procedures. Specific tools and expertise need to be identified. Questions that need to be asked include: Is the targeted end-user an individual or a group? What expertise is required? Hence, engineers should go through a brainstorming process. It is important for software engineers to develop a product that is not only useful but also user-friendly, so that it is easily marketable.

Theory

We need a good theory that is grounded on a viable philosophy so that we can adopt appropriate approaches for software development. The applications of software range from the simple to the complex. In almost every sector of modern life, software has revolutionized everyday living, helping to solve problems and making our lives safer and more comfortable.

System Engineering Process

State the problem

One of the most vital functions of system engineering is the identification of the problem. It describes the functions that a system must perform to satisfy the needs of the target clients. Stating the problem also helps engineers to identify what functions their system requires. In system engineering, it is important to know what needs to be done to the system to arrive at the desirable outcome. System engineers need to be aware of their target clients' needs and expectations before they can develop software that is marketable.

Explore alternatives: In order to explore alternatives, engineers have to evaluate risks and costs before they find an alternative to the problem. They also look at the performance required by the system.

Model the system: Replicating the system has many advantages in system engineering. Among others, the system requirements and basic structure are already available. As such, adapting to a pre-existing model is less expensive.

Assimilate: In this phase of system process, engineers try to bring all the system components together, and a holistic approach is adopted. In order to accomplish system integration, there must be synchronization and communication.

Integrate the system: At this stage, engineers test out the system, applying the software that has been developed. This stage is very important because the system has to complete a predetermined job, i.e. the system must function according to its specifications.

Evaluate performance: A comprehensive evaluation of the system is carried out. At this point, the system must work as planned. Its performance must be measurable in terms of its ability to meet the client's needs. Once it is measurable, it is easy to determine if the predetermined benchmark has been reached, and to make further improvements if necessary.

Re-evaluation: This is about continuous measurement and evaluation to ensure consistent performance, and to consider constant improvement to the system.

System Engineering Life Cycle

Requirement: System requirement defines what our system should do. System requirement is about the functions of a system and how it should perform. It also outlines the essential properties of the system. There are three main requirements attached to the system:

Abstract Requirement: It is about the essential functions of the system in abstract form. At this stage, the functions of the system are very limited compared with the subsystem. In the subsystem, more detailed functions are attached.

System Properties: System properties are idle, and have no important functions to play in this phase. System properties deal with risks, safety and performance at the requirement stage but become more important upon reaching the subsystem level.

Specifications: At the specification level, the system should not show any sign of weakness or irregularities. The system should function according to its specifications. It should be user-friendly. For instance, the system should not generate too much information for the receiver so that he/she is unable to understand it. The overall aim of the system is to adopt a holistic approach to achieve the objectives of the system.

System Design: In order to design a viable system, engineers partition the system into subsystems so that their functions are distinct. Moreover, reusing subsystems helps to maximize the functions of an existing system. If it is not possible, engineers will buy another subsystem to add extra functionality to the existing system. And if this cannot be done, engineers will modify the existing subsystem.

Only when the engineers are happy with the modification will they build the subsystem internally. Hence, engineers must be familiar with the interaction and communication between the system and subsystems.

System Design Roadmap

Some of the most important activities in designing a system are as follows:

Partition: Make partitions in order to organize those with similar functions so that you can have alternatives.

Subsystems: Then identify the subsystems so that they can collectively or individually fit into the subsystem.

Assign to the subsystem: Assign whatever functions are suitable to the subsystems. If it is accurate, use it; otherwise, modify it.

Assign functions to subsystem: Specify what functions your subsystem should perform. Identify their relationships so that you know what each system does.

System decommissioning: Determine the life cycle of a system. If there is no more support or funding for the project, it has to be abandoned. Sometimes it is abandoned intentionally because no more evaluation is needed. Once the system is successfully implemented, it is recorded for future reference. Hence, as system engineering requires many different types of expertise, teamwork is critical for the design and development of good software.

What is Quality Software?

There are various definitions of quality software. According to Garvin (1984), quality is something that we can visualize but unable to define with accuracy. According to him, a product has satisfactory quality if it does what it is supposed to do. Quality is often associated with the price of a product, although this does not necessarily imply that reasonably priced products are inferior in quality or vice versa. In addition, discussions on software quality should also take into consideration the following:

Extent of faults, failures and defects, ranging from the minor to the disastrous
External view and internal view
Design and code, maintenance
Reliability, efficiency, integrity
Usability, maintainability
Flexibility, portability, reusability,

There are different views of what quality software should be. According to Matias (2009), if the software passes the process of rigorous testing, it should work according to its specifications. It is vital that developers continue to improve their products in line with changing needs. Victor's (2005) view of quality software depends on customer satisfaction. If a customer is satisfied with a product, then it is a quality product. Software engineering has improved tremendously over the years. Software developers are able to avail themselves of the latest techniques and state of the art technologies to reshape their products. Nevertheless, despite much advancement in software engineering, the issues pertaining to quality remain unresolved. Many scholars are of the view that the quality of software is very complex, and therefore rigorous investigation must be carried out in order to determine what constitutes quality. Kurilovas (2010) suggests the following points for consideration:

1. Achievement of objectives and goals.
2. Standards must be specified: Accurate **processes** such as pre-defined stages must be in place in order to create an industry-wide agreement.
3. Meeting of specific clients' expectations.
4. Adherence to software development ethics.
5. Value and cost effectiveness
6. Quality improvement opportunity
7. Models of software quality
8. Quality assurance to customers
9. Completion of validation and verification.
10. Conduct of reviews and implementation of suggestions.
11. Quality of Application.
12. Identification of faulty characteristics (minor, major, disaster).
13. Quality management techniques.
14. Quality measurement.

According to Sommerville (2007), if you want to manage your software product efficiently, there are some guidelines to follow:

1. You must have quality assurance measurements such as a system design roadmap that ensures your software meet with international standards.
2. You must have a logical and a feasible plan of action to guide you in the development of a quality product.
3. You must have established rules and regulations governing software quality. This will guide developers to process their software efficiently.

Therefore, a quality management plan not only ensures quality but also helps developers use the check and balance concept to improve the quality of their software. It also ensures that the software has followed organizational standards, and has met its goals and objectives. A quality management plan is divided into two phases; one is handled by quality assurance experts and the other, quality development experts. These experts are assigned different tasks so that the process of software development is viewed through different lenses. Both teams report the problems, if any, to senior management for further action. Once a product is developed, it is rigorously investigated for quality performance. If the team finds that the quality is not up to the mark, the product goes back again to the quality development experts for rectification of the problem. Only when the quality of the product is deemed satisfactory will it be allowed to proceed to the standard processing level.

How do we Measure Quality?

It is important to assess the quality of the software before it is in the market. Further, if we want to measure and improve the quality of our product, we have to refer to software models for quality assurance and measurement. For example, how do you measure a good design for its quality? We can use software models as our guide.
According to Francisca (2003), the following are some quality measurement techniques:

FUNCTION:	Suitability, Accuracy, Compliance, Security
RELIABILITY:	Maturity, Fault Tolerance, Compliance, Recoverability
USEABILITY:	Understanding, Operational, Compliance
EFFECIENCY:	Time Saver, Resource Saver, Compliance
MAINTANENCE:	Stability, Testability, Compliance
PORTABILITY:	Replace Ability, Adaptability, Installability, Compliance

Conclusion

As evident from the above discussion, to develop good software, we have to identify the needs or problems of the target clients, analyze the problems, and finally create solutions for the problems. Quality supersedes quantity in software development. It is crucial to follow a system design roadmap to ensure that quality software is developed. Rapid technological advancement coupled with new problems and demands of modern living mean that system engineers have to constantly come up with software that will satisfy their clients' needs.

References

Cagle, W. Marsha. (2010). Effective Software Engineering Leadership for Development Programs. *ProQuest LLC, D.M. Dissertation, University of Phoenix*

Cristobal, J.; Merino, J.; Navarro, A.; Peralta, M.; Roldan, Y.; Silveira, R. M. (2011). Software Engineering Infrastructure in a Large Virtual Campus. Interactive Technology and Smart Education, v8 n3 p172-185

Daugherty, J.; Custer, Rodney L.. (2012) Secondary Level Engineering Professional Development: Content, Pedagogy, and Challenges. *International Journal of Technology and Design Education, v22 n1 p51-64*

Francisca Losavio. (2003).Quality Characteristics for Software Architecture, *Journal of Object Technology*, vol. 2, no. 2, pp. 133-150.

Garvin. D. (1984). What Does Product Quality Means? Sloan Management Review, 26, No. 1, 24-43

Golden, E. (2010). Early-Stage Software Design for Usability. ProQuest LLC, Ph.D. Dissertation, Carnegie Mellon University.

Kurilovas, E.; Dagiene, V. (2009). Learning Objects and Virtual Learning Environments Technical Evaluation Criteria. Electronic Journal of e-Learning, v7 n2 p127-136

Kurilovas, E.; Dagiene, V.(2010). Multiple Criteria Evaluation of Quality and Optimisation of e-Learning System Components. Electronic Journal of e-Learning, v8 n2 p141-151

Matias. M. (2009). Software Process improvement: continuous Integration and Testing for Web Application Development. University of Tampere, Department of Computer Sciences, Thesis.

Shari, L. Pfleeger, J. A, Software Engineering Theory and Practice, 3rd Ed. Upper Saddler, River, New Jersey, 2006

Sommerville, L. (2007). Software Engineering, 8 ed. Pearson Education Limited, Edinburgh Gate, England

Victor, E. Sower, Frank, K. F. (2005). There is more to quality Than Continuous Improvement: Listening to Plato. Houston State University

Walt, S. (2001). Process Models in Software Engineering, Institute for Software Research, University of California, Irvine

Watts. S. Humphrey, (2008). The Software Quality Challenge, Journal of Defense Software Engineering. The Software Engineering Institute.