Software Engineering Basics & the Algorithmic DocM Model for Software development

Azeem Uddin
B.tech (Computer Science & Engineering), Galgotias University, Greater Noida, Uttar Pradesh, India

***Abstract*- Software is a computer program that when executed provide desired features, functions & performance. Software engineering is a layered technology with quality, process, methods & tools as it's crucial layers. The development of a software requires proper documentation, analysis, planning, supervision & management. Software engineering enables us to build complex systems in a timely manner. It is the application of engineering to the development of software in a systematic method. Hence, this paper provides a general description about Software engineering & it's basics. Software is developed with the help of SDLC framework i.e. Software development lifecycle that has many phases like requirement gathering, feasibility study, system analysis, software design, coding, testing etc. SDLC is a sequence of activities that lead to the production of a software product. So, this paper also aims to introduce a new brand SDLC Model i.e. THE ALGORITHMIC DOCUMENT MAINTENANCE MODEL. This Model is described in a thorough manner covering phases, advantages, significance & algorithmic approach.**

Index Terms- Software Engineering, Software development lifecycle (SDLC), Fundamentals of Software Engineering, The Algorithmic DocM Model.

INTRODUCTION

Software engineering is the combination of software & engineering. Software refers to a program or set of programs that when executed provide desired features, functions & performance whereas engineering is all about developing new products using well-defined scientific methods & principles. So, combining both the aspects, Software engineering may be defined as an engineering branch that is associated with the development of a software product. The outcome of the software development process is an efficient & reliable software product. Functionality, reliability, efficiency, usability, maintainability, portability, robustness & integrity are some of the main software characteristics. Software is developed with the help of SDLC framework i.e. Software development lifecycle that has many phases like requirement gathering, feasibility study, system analysis, software design, coding, testing etc. SDLC is a sequence of activities that lead to the production of a software product. A software consumes resources, budget & time. Development of a flexible & efficient software product requires satisfactory software project management. Software development can be divided into two parts- Software creation & Software project management. Software project management is the art & science of planning & leading software projects.

Software product is influenced by four P's i.e. People, Project, Process & Product. There are many software development lifecycle models like waterfall model, prototype model, spiral model, iterative enhancement model & many more.

Software engineering: A layered technology

Software engineering is an engineering branch that is associated with the development of a software product. There are four layers that act as a base for software engineering. Due to this reason, software engineering is called as a layered technology. All the four layers of software engineering are briefly summarised below-

1. Quality: Quality refers to the fitness for purpose. It is a subjective aspect. There are many software quality attributes that software must possess. In fact, the bedrock that supports software engineering is quality focus.

2. Process: Process is one of the four P's on which software project management focuses. When a software program is executed, it becomes a process.

It is a framework that must be established for effective delivery of software.

3. Methods: Focuses on HOW to build the software. Each method consists of multiple tasks like requirement analysis, testing, maintenance etc.

4. Tools: Tools are used to build up the software. Tools provide automated/semi-automated support for process & methods. For instance, CASE(Computer-aided software engineering) tools are used to design & implement applications.

Principles of software engineering

Principles are basic ideas or rules that explain or control how something happens or works. Principles in software engineering are necessary for uniformity, discipline & output. Five principles of software engineering are described below-

1. Think through the problem completely before you try to implement a solution: This principle states that firstly, one should understand the problem thoroughly, then only he/she should start implementing the solution because proper understanding of the problem statement & analysation process are necessary to find the appropriate solution.
2. Divide & Conquer: Divide & Conquer approach states that the entire problem must be divided into sub-tasks for proper management. It is also called modularization. This division of the whole work increases the reliability, concurrent execution & maintainability.
3. Keep it simple: It means that there is no need to make the software process complicated. Developers, project managers & the people who are associated with the software product must keep the software process simple & easy to understand & implement.
4. What you produce, others will consume: It is totally true that software products are made for the users or customers. Developers make them in accordance with the requirements & expectations of the users. Hence, a software must be user-friendly. User/Customer must be able to use the software with ease.
5. Learn especially from your mistakes: All humans make mistakes. There is nothing wrong in doing mistakes but one should learn from his/her mistakes. The same approach follows with regard to software as well. The developers must learn from their mistakes, their ultimate goal must be to produce a quality software product & they should fix the bugs/errors with great attentiveness

The Algorithmic DocM Model for software development

The DocM Model stands for Document Maintenance Model. This model refers to a systematic & planned approach for the development of a software. The creation of a detailed document is the most crucial aspect of this model.

BASIC STEPS (Phases) INVOLVED IN THE DocM MODEL

There are many steps involved in the DocM model that must be followed sequentially & with great management. The various steps are explained below-

1. COMMUNICATION: The very first phase is communication phase. In this phase, the customer contacts or approaches the service provider or the developer to express his/her desire of software product. The user or customer also tries to negotiate the terms at this step.

2. REQUIREMENT GATHERING: At this step, requirements are gathered from the customer. It focuses on WHAT not HOW. Discussion is carried out between customer & developer. It is an activity that helps to understand what problem has to be solved & what customers expect from the software. The foundation of this phase is effective communication. There are many requirement elicitation methods like interviews, brainstorming sessions, Facilitated application specification technique (FAST) etc. This step involves the practice of collecting the requirements from users, customers & other stakeholders.

3. FEASIBILITY STUDY: In this phase, it is checked that whether the project is feasible (Workable) or not. It concentrates on the operational feasibility, technical feasibility & economic feasibility. At last, a feasibility report is created that specifies whether the project is practically possible or

not. This step is just an assessment of the practicality of a proposed system or project.

4. DCT analysis phase: Here, DCT stands for Design, Coding & Testing. This phase is not about the actual implementation, it is about the analysis & forming conclusions. The development team focuses on-
a. DESIGN: It is all about the modelling techniques like use case approach, Entity relationship diagram, data flow diagram etc. The developer thinks which model must be used to portray the functionalities, behaviour & structure of the software system.
b. CODING: The development team decides which programming language is the most suitable for the proposed system. Actual coding is not started but a general idea is set regarding the programming methodologies. But, pseudo code or rough estimations are possible.
c. TESTING: Testing is the combination of verification & validation. But, at this step actual strict testing is not started. Only the test criteria & test case plan is created. More emphasis is given to the principles & objectives of testing with regard to the software project.
Hence, this DCT phase is only about analysis, planning & estimation. It is not about the actual implementation.

5. Writing Documentation: After the DCT analysis, there comes the role of proper documentation. A detailed document is created by the development team that comprises the following information-
a. Software requirements
b. Requirement analysis report
c. Feasibility report
d. Design details
e. Pseudo code
f. Test strategy & test plan
g. Characteristics of the software product
So, this document comprises of all the necessary details about the software product. The creation of this document is a time consuming process because all the details about requirements, design, coding & testing are included in it. The ultimate objective at this step is to generate a formal document that is understandable to users.

6. Document approval or rejection phase: After creating a final document, the developers show it to the customer for customer satisfaction & feedback. If the customer is satisfied with the document or he/she approves the document, then the developers can start the actual implementation or working from the design phase to the maintenance phase. But, if the document is not in accordance with the customer's expectations or the customer rejects the document, then a new document is created & again it will be shown to the customer until or unless the customer gets satisfied. This process will be iterated till the creation of a satisfactory document & after that it should be verified by the customer. Once it is verified by the customer, the actual working is started using required tools & methods.

7. Delivery of software product to the customer: Once the verification is complete, the developers try their level best to build a high quality software. When ready, this software is delivered to the customer & further maintenance is taken into account.

ALGORITHMIC REPRESENTATION OF THE DocM Model
START
a.Communication
b. Requirement gathering
c. Feasibility study
d. DCT analysis
e. Making of a formal Document
f. Verification of the document by the customer (User)
g. If verified, then START ACTUAL WORKING (IMPLEMENTATION) from design phase to testing phase.
h. If not verified, then make a new Document & iterate the process till the customer is satisfied.
i. Delivery of software product to the customer & maintenance.
STOP

Advantages of the Algorithmic DocM Model
1. Algorithmic approach: This DocM model is algorithmic in nature. It specifies the step by step procedure for the development of a software product. It focuses on input, output, effectiveness, generality & ease of use.
2. Document-oriented: This model is document-oriented as a detailed document is created & shown to the customer before actual implementation of the

software. Documentation is the written & retained record of events that is very crucial in this world where there are fickle-minded customers. Hence, proper documentation improves the reliability, maintainability & accuracy of the software product.

3. Customer Involvement: Customers (Users) are the people who greatly influence the software building process. So, in this model, there is more customer involvement as after the creation of document, the document is disclosed to the customers for their feedback. It is done so as to gain the customer confidence & to check the completeness, consistency & correctness of the document.

4. Understanding System: Since a general idea (Outline), behaviour, functionality & structure of the software product is displayed through the easy explanation & modelling techniques, the user get a better understanding of the system.

Significance of the Algorithmic DocM Model

The Algorithmic DocM Model can be of utmost importance to develop a user-friendly software. This model emphasises on customer feedback & satisfaction. It is simple & easy to understand as a formal document is maintained in this process. This document plays a vital role throughout the software development cycle. It is an ALL IN ONE document that includes details regarding design, coding, testing etc. There is no need for a separate software requirement specification document. This model is Document-dependent. Written communication skills & speaking skills are used by the developers to develop the software using this algorithmic model. In this model, a document is refined again & again according to customer instructions & feedback, so this model can also handle change. After the delivery of software also, the document involved in this model is refined & maintained with operation & maintenance information.

In short, the Algorithmic DocM Model is a systematic & planned way of building a software product.

CONCLUSION

Software is an intangible product. The development of software requires proper documentation, analysis, planning, supervision & management. Software engineering enables us to build complex systems in a timely manner. It is the application of engineering to the development of software in a systematic method. The power of software engineering must not be underestimated. With the help of software engineering, high quality software can be produced. Moreover, an effective software project management can also help transfer the product from zero to hero. Hence, this paper has explored the software engineering basics & an efficacious methodology/model for the development of robust & reliable software.

In this high tech world where user is so fickle-minded, building high quality software is not a cakewalk. But, if correct strategies & managerial aspects are followed, then quality software can surely be maintained. Even more fundamentally, it is important to recognize the need of software engineering. The Algorithmic Document Maintenance model described in the end sums this up neatly.

ACKNOWLEDGEMENT

First & foremost, praises & thanks to the God for his shower of blessings throughout my research work to complete the research successfully.

I would like to express my deep & sincere gratitude to my parents for their love, prayers, care & sacrifices for educating & preparing me for my future. I would also like to express my gratefulness to all my academic colleagues & friends for their constant encouragement.

Finally, my special thanks go to all the people who have supported & encouraged me to complete the research work directly or indirectly.

REFERENCES

[1] Rajib Mall, Fundamentals of Software Engineering, PHI Publication.

[2] Pankaj Jalote, Software Engineering, Wiley.

[3] M. Cotterell, Software Project Management, Tata McGraw-Hill Publication.