GOOD PROFESSIONAL PRACTICES FOR DEVELOPING GOOD SOFTWARE

Niral Verma
Assistant Professor
Department of IT/CS
Pillai HOC College of Arts Science and Commerce, Rasayani
nirali.verma0902@gmail.com

Remya O
Assistant Professor
Department of IT/CS
Pillai HOC College of Arts Science and Commerce, Rasayani
remyaomanakuttan90@gmail.com

Abstract— To develop a good quality software lot of experience and coding skills is required. Thus, beginner or inexperienced developers suffers lot. In this general rule to help in this important context paper have been highlighted. It presents the directions and good professional practices of developing good software regardless of software realm(whether for mobile, desktop, web or embedded), size and complexity. Good practices are organized in categories to ease the process of contemplate them while developing software.
Also many recommendations are given in different categories which can be considered by software developers.

Index Terms—Java, C#, monolithic, microservices, AUI, SDLC, CLI, ZUI, IDE, VUI, GUI

1 INTRODUCTION

Writing software is the art which is a process of converts what is in mind into reality. This process produces software-based solutions for real-world problems which we face in our everyday life.We use software everywhere like in cars, aircrafts, and many others.For developing software we use programming languages such as Java, C#, etc for coding. Also, different types of models, methods, and tools are used to manage this development.Software that serves our different types of needs is not written equally. Many software developers who are involved in the Software development process suffer from the issue of writing good software or produced.It is crucial to develop good software. To address this mentioned issues, this paper provide helps in this context as best practices and recommendations for writing a good software. This paper is written to be used as a guide to support software developers to write good software. As well, a set of qualities, standards, rules, and practices which are closely related to the principles of good design and engineering but are not limited to any programming language, software domain, or the educational background of whom is writing the software has been gathered, organized, and presented.

2 LITERATURE REVIEW

There are many articles for writing software which are specific to certain programming languages and focus only on few aspects of writing software.The software combines creativity and imagination It is what mostly people use in their practical lives. We cannot imagine without different types of software products which are used in industry, healthcare, military, transportation, etc. Software products have special characteristics that differentiate them from non-software products. In developed software products, new requirements come in all the time so we don't have to develop it from scratch; changes are applied on the same product. These products do not tear off after too much use. Some software delivers its functionality in a right way but we cannot consider them as good software. A single mistake in development of software may suddenly fall down the work of much years.The growth and improvement in the realm of software is growing fastly. [2]Software Engineering is the field that applies the discipline and structured approach for sofware development.It is a field through which we can build a quality product in defined time. There are new innovations in this field regularly which includes development of new software applications, frameworks, and technologies. The one who is studying software requires studying other fields in depth.[7] Developing software products may cost nothing in terms of development costs in many cases. We cannot consider software as a program as Software is developed by a group of people called team which is not in writing programs. Software is used to identify programs, data and other files which are used to complete a task in computer or other devices whereas programs are the instructions which are executed by machines.

3 Proposed Wrok

For producing a quality software the approach and recomendations has been classified in to three categories a) Before Developing software b)While developing software, and c) After development of software.[1]Consideration of reccomendations before starting the process of writing software will help to develop a good software.
Before development of software give a careful thought on the following:-

1) Make a good team

[3]When the software is small it casn be developed by an individual but when the software is large it needs a team.Team is important for planning, organizing, monitoring and controlling the developmemt process. Good team leads to a good product. Team members should know about their roles and responsibilitie. Proper vision and developing a good software should be a goal for every member of team . Software team not only contains developer but it also has other members who are not developer but perform important functions for development of good software. There should be a clarity in expressing ideas among all members of team. Timely meetings and group discussions plays vital role.The team should be technically skilled, team members should work collabratively. The team should have good leader as leader plans and manages task among the team members. The size, skill of a team should match the task.

2) Work with Experts

Working in touch with professional or experts helps to build good software which can be achivied by building a good team. This can also be aschieved by training courses, workshops or confrences, websites, blog pages, youtube channels, books, research articles etc.

3) Understand the software

There are different categories of software like web software, mobile software, scientific software, embedded systems, scientific software, business software, Engineering software and Desktop software. Understanding attributes of software of specific domain helps to create a quality software.

4) Select development, architecture, programming[8]

For writing a good software proper selection of software model is very important. Selection of Software model depends on many factors such as team size, complexity, duration, domain etc. There are different models such as waterfall, incremental, agile, prototype and spiral, with its own usage, advantages and disadvantages. For example:- Waterfall Model, in this the task go sequentially in multiple phases. Waterfall model is also called as classical model or linear sequential model. The six phases of this model are

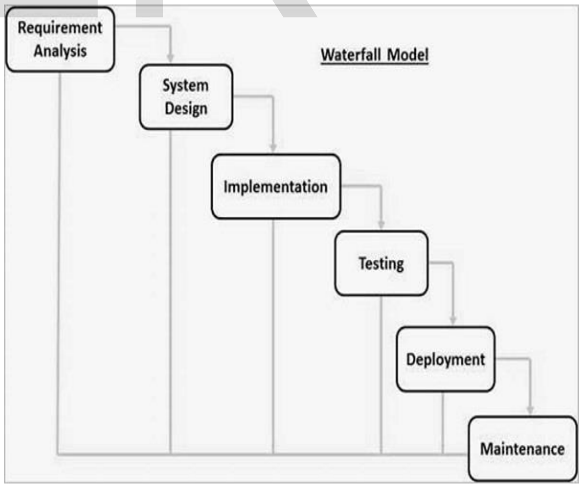
1. Requirement Gatehering and Analysis
2. System Design
3. Implementation
4. Testing
5. Deployment
6. Maintenance

The advantages of this model are:-

- Simple to understand and use
- Provides clear separation of task

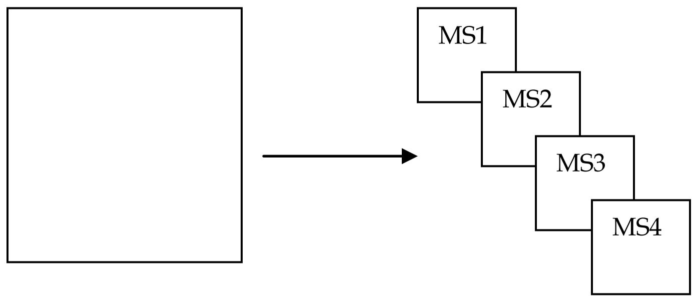
The disadvantages of this model are:-

- It does not consider user feedback
- It is time consuming.



Waterfall model is used as basis for other software development models. Selection of appropriate model is very important as it may affect the software product and team

[4[Architecture means how the structural elements and functional components are organized in software product and how they are communicating with each other. While selecting a software architecture one should be careful as it affects quality. There are various architecture. For example traditional monolithic



There are many languages which are used to develop distinguish types of software. Many languages have common characteristics and thus target common domain such as web, desktop etc. Many have unique characteristics and thus target particular domain. Language has direct impact on quality and functionality of software. Developer should deeply go through all the features, concepts of language to use it efficiently and effectively.

5) Select leading IDE and user interface

Integrated Development Environment has code, editor, interpreter/compiler, and debugger and built in utilities which hide confusion and ease the software development. Advanced IDE increases the productivity. There are different IDEs having its own set of features supporting different programming language. Like in NetBeans IDE user can measure CPU time, RAM allocation, execution time of all functions and many more.

Good software has good code and good user interface to its user to interact with its functionality. Software developers should take this point into account while coding. User interface are divided into many categories:-

Command Line Interface(CLI) is a simplest one among other user interface in which user interact by software by typing commands in a screen using the keyboard. Graphical User Interface(GUI)has friendly mechanism in which graphical components are used for interactions. Zoomable User Interface(ZUI) is a special type of GUI in which user is able to see more detail by changing scale of viewed area. Voice User Interface(VUI) helps user interact through voice or speech command. In Activity User Interface(AUI) user interact through gestures originated from hands or face.

User interface should be clear, concise, attractive, efficient, forgiving, feedback and speed learning.

6) Work in good environment

Many studies shows that physical environment affect code quality and productivity. Lighting, Colouring, Ventilation, Air conditioning, Adjustability are the various elements of physical environment that affect code quality.

While Writing Software

Best practices which can be followed for writing good quality software are:-

1. Avoid quick coding as it has not good impact on quality of work.
2. Use proper indentation.
3. Follow coding level by using spaces to indent code, variable name should be self explained, easy to remember. Use proper naming conventions for classes, functions, interface, namespaces.
4. Simplify coding is important. Create simple classes having well defined functionalities.
5. Relationship between classes should be simple and understandable.
6. Reuse existing code
7. Avoid code duplication.
8. Use comments for description of code which is a very important practice.
9. Minimize coding. Software should have more cohesion and less coupling.

After writing software

After writing software developer should do the following:-

1. Review his code which he has written
2. [6]Testing insures that the code is without defects or undesired functionalities. Different types of testing are:- White-box testing to test the internal structure, Black Box Testing to test the functionalities, Gray Box Testing combines both to test internal structure and functionalities.
3. Maintenance of code which is done after real installation and deployment of software. Corrective maintenance, Perfective maintenance, Adaptive maintenance, Preventive mainte-

nance are the different types used to maintain code.

4. Make searchable code. The code which is discovered and accessed easily is a good code. Therefore, developers should make their code accessible.
5. [5]Create user manual.

- Good software should have good user manual. User manual usually have different sections like
- Title which includes software name, version, copyright, manual created.
- Content helps in navigation of manual .
- Introduction gives the overview of software.
- Requirement gives the list of hardware and software required.
- Frequently Asked Questions.
- Installation having set of instructions needed for installing software.
- Uninstallation having instructions for uninstalling the application.
- Using tells about how to use the software.
- Troubleshooting tells about the list of error which can occur and how to handle them.
- Contacts tells about the where to find technical help and contact details.
- Glossary tells about the term which are not familiar

4. CONCLUSION

This paper has the classification of very significant best practices, suggestions, elements that software developer can follow for producing a good quality software. There are many factors that targets the technical aspects. Everything given on this paper will help developer to use as guide to write good software regardless of software domain whether for desktop, mobile, etc.

5. REFRENCES

1.http://www.differencebetween.net/technology/difference-between-software-and-program
2.https://whatis.techtarget.com/definition/software-engineering
3.https://cybercraftinc.com/blog/roles-duties-goals-and-kpis-software-development-team-organization
4.https://dzone.com/articles/monolithic-to-microservices
5.https://www.wikihow.com/Create-a-User-Manual
6.https://en.wikipedia.org/wiki/Software_testing
7.https://www.estimancy.com/en/2018/11/05/software-requirements-8-best-practices-to-write-them}
8.https://opensource.com/article/17/5/30-best-practices-software-development-and-testing