International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6
Article Received: *25 March 2023* ***Revised:*** *12 April 2023* ***Accepted:*** *30 May 2023*

A Study of Reusability of Component on Software Engineering

Praveen Kumar
Research Scholar,
Computer Sciences & Application
Dr. A.P.J. Abdul Kalam University, Indore
Praveenkr1@gmail.com

Dr. Kailash Patidar
Research Supervisor,
Computer Sciences & Application
Dr. A.P.J. Abdul Kalam University, Indore
kailashpatidar123@gmail.com

Abstract : The process of creating new software is equal parts art and science. It's the process of creating something that meets requirements while staying within budget and deadline restrictions. First coined in the 1960s during seminars hosted by the NATO Science Committee in response to the so-called "Software Crisis," the term "Software Engineering" has since become common parlance in the IT industry. For its focus on modularity and reusability, Component-Based Software Engineering (CBSE) stands out among the numerous subfields of Software Engineering. CBSE is a sub-paradigm of software engineering that places an emphasis on software artifact reuse and the reusability lifecycle. CBSE strongly advocates the idea of "buy, don't build." The Component-Based Software Development (CBSD) method streamlines the development of software systems by choosing and integrating suitable components from a library of prebuilt, reusable (off-the-shelf) software work-products. CBSD recommends using pre-made, context-independent, and diverse parts while developing software. Component-Based Software solutions are constructed by integrating reusable, pre-existing, and new components linked through error-free interfaces.

Keywords : *Reusability, Software Engineering, Component-Based Software Solutions, Software Systems.*

I. INTRODUCTION

Over the last three to four decades, the software business has expanded greatly. Software is essential for many different types of vehicles, embedded systems, internet commerce, and service businesses such as banking, telephones, and hotels. The three most crucial aspects of any software development process are budget, schedule, and quality. The majority of the total cost is attributable to the use of highly skilled professionals and the ongoing development of relevant technologies. Planning a timeline may be quite helpful for software development initiatives. Timely completion of software development is crucial to the success of any project. Many software projects have failed to meet their deadlines in the past, which is not a promising sign for the industry as a whole Today, software quality is a core value, right alongside budget and timeline, for developing top-notch software products.

Multiple software companies have been crying "software crisis" since the 1980s due to problems with the software development process. Robert Glass has said, "I look at my failure tales and see exception reporting, spectacular failures in the middle of numerous triumphs, a cup that is [now] almost full." While progress has been achieved in the software sector, there is always opportunity for enhancements. One of the major challenges in software development is understanding the current body of information and then applying the relevant domain logic to it. Ignoring the efforts of a person to improve the software development process will not be compensated for by a fast data center or cutting-edge hardware developments. Open-source, like SaaS, is the new approach for the software industry to meet customer demand, but it needs a higher skill level from employees.

The software development process has advanced with technology to remain competitive in today's software business market. The software development process has too many long-held assumptions that have been reinforced by experience and observation. However, not all types of recursive reuse are advantageous in the long term, as shown by the cost-benefit analysis of the development process.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6
***Article Received:** 25 March 2023 **Revised:** 12 April 2023 **Accepted:** 30 May 2023*

While some notable companies have been successful in using an improvement process approach, others have attempted and failed because they failed to properly appreciate the benefits. Despite the upgrade's negative impact on profits, many companies persist in investing in time-consuming software engineering.

Several software businesses have published programs on a variety of platforms in preparation for the third decade of the 21st century. Some have developed in novel methods to further strengthen the stability and reliability of the system/application software. They will be used as soon as feasible since they were delivered on time and improved in quality. Many big companies have chosen a software development process that puts a premium on technical and project management duties, and this approach is considered the "standard (varies on the basis of individual business policy)". Many firms, however, persist in using the trial-and-error technique of software development.

As a result, success in today's competitive software business market requires both kinds of organization to engage in daily feedback-based software process improvement (SPI) in the direction of defining standard policy (legal or digital) or, alternatively, throughout the whole software development process.

II. SOFTWARE PROCESS IMPROVEMENT

The software development life cycle is broken down into the following stages when using SPI:

(a) Key software development process indicators should be reviewed and updated.

(b) Determine the value of the method's useful parameters by determining how challenging it is to implement in practice.

(c) After doing the aforementioned analysis, the most effective method for making advancement may be implemented. The traditional approach to software development is reimagined in light of the SPI technique, which is more flexible, adaptive, and reliable. The method for improving software is worthwhile in the long run, despite the time and effort required. Process enhancements may lead to a reduction in the number of useful software parameters. Less money will be spent on software maintenance, less time will be wasted on quality issues, and less hidden costs will be incurred as a result of software delivery delays. The method for improving software takes a long time and requires a lot of work, but it is worth it in the end. Therefore, less parameters in software are needed for more efficient operations. It also reduces the time and effort spent correcting quality issues, the expense of software maintenance, the risk of passing errors on to consumers, and the indirect costs of delivering software late.

III. APPLICATION DEVELOPMENT

The phrase "software engineering" is used to describe a set of procedures where an engineering approach is used to developing and maintaining software. By "technology," we refer to the whole spectrum from abstract concepts to operational structures. The IEEE defines software engineering as follows:

To put it simply, Software Engineering is the application of engineering principles and practices to the process of creating, deploying, and maintaining software. [2]

In this regard, Philippe Kruchten suggests that software engineering may be distinguished from Civil, Electrical, and Mechanical engineering in the following ways: -

- Since there aren't many widely accepted theories on how to go about creating software, theoretical Software Engineering research may be difficult.
- Changes to software are encouraged despite the difficulty of predicting their impact.
- Emerging technologies cannot be adequately evaluated due to the rapidity with which they are evolving.
- It is important to remember that fixing issues, installing updates, and reworking the system are not accounted for in the initial software cost estimate.

Some of the current problems in software engineering include novel approaches to software development, Open Source Software (OSS), Commercial Off-The-Shelf (COTS), and a more methodical synthesis and aggregation of multinational domains.

IV. AN OVERVIEW OF COMPONENT-BASED SOFTWARE DEVELOPMENT

The phrase "component-based software engineering" (CBSE) first appeared around the turn of the century. This technique was created because object-oriented reuse analysis has been greeted with dissatisfaction by the software development community. Integration of loosely coupled software components requires their specification, selection, implementation, retrieval, and incorporation into applications.

In-House, Outsource, and Cots Methods for CBSE

Component-based Software Engineering allows for the usage of both custom and external software components. Both COTS and OSS (Open Source Software) refer to third-party software components. Accessible in large numbers,

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6
Article Received: *25 March 2023* ***Revised:*** *12 April 2023* ***Accepted:*** *30 May 2023*

commercial off-the-shelf (COTS) products are notoriously difficult to understand owing to their commercial nature.

The source code for open-source software (OSS) components, on the other hand, is usually available to anybody who wants to have a look Commercial off-the-shelf (COTS) software has third-party support for even earlier versions, and its components are normally accessible for the three most current editions. As documented by (Basili and Boehm, 2001).

The Advantages of Partitioned Software Development

Component-based Low-Development-Expense Software Engineering is useful for reducing expenditures on new software because of its emphasis on reusability. The need to develop software from scratch is therefore obviated. As a result, less garbage is produced, which in turn lowers expenses.

Time to Market is Decreased

CBSE reduces development time by testing and debugging all software components in advance of writing code.

Therefore, the time spent testing and fixing bugs in the program is minimized when employing software components.

Software Complexities That Are Simple To Administer

To simplify the complexities

Higher Standards for the Software Engineering Process

In the course of reusing software parts, there are several chances to catch and fix errors. This is why our software is so trustworthy.

5% less Defectiveness

By dividing the total number of bugs by the total number of lines of code, we can get the defect density. CBSE employs reusable software components that have already been tested and configured, leading in applications with a reduced defect density.

V. HYDROLOGY AS AN EXAMPLE OF A DOMAIN

These investigations will be put to use in the domain of Hydrology. The hydrological cycle is at the heart of hydrology, the scientific study of water's endless cycle on Earth and in its environs. The world's water supply are managed for the greater benefit with the help of hydrologists. The water supply on Earth is affected by shifting distribution practices, climate change, and the water cycle. The following are some of the ways in which humans affect water supplies:

- Pumping ground water
- Building dams
- Deforestation
- Use of excessive fertilizers
- Irrigation of land

There are a plethora of positive and negative human activities that impact water quality and circulation on Earth. The following are only a few of the numerous uses for Hydrology information:

- Flood Control
- Hydro power generation
- Irrigation
- Water Distribution urban and rural area
- Pilgrimages (Badhrinath, Kedarnath, Yamunotri, Gangotri, Haridwar)
- Snow Gagging, / Glacier
- Dam Water Management
- Ecology
- Industry demand

Those working in science, research, planning, agriculture, administration, management, engineering, local government, county government, and the armed forces all depend on hydrological data to carry out the aforementioned social good activities.

VI. APPLICATIONS IN THE DEFINED FIELD

Hydrology software is used by a wide variety of professionals, including hydrologists, water managers, researchers, scientists, and academics. Software in this area often includes tools for data integration, GIS mapping, scenario planning, optimization, statistical analysis, report production, research project management, and workflow control.

Most of the software in this area is used to encode physical laws as mathematical algorithms, integrate with geographic information systems (GIS), computer-aided design (CAD), and two- and three-dimensional imagery. Commercial software with hydrology-related features is provided by a number of businesses, including Deltares, DHI software, Innovyze, Bentley, and others. [4] There is a present lack of

significant software use in the hydrological area. It's important to learn more about software reuse so that more software problems may be solved.

VII. CONCLUSION

Component-based software development, which emphasizes separation of concerns, has recently spurred a revolution in the software industry. Software is built from smaller, more specialized pieces called modules. In software, the whole is greater than the sum of its parts.

Maintaining high standards of quality throughout growth is essential. The development team requires precise knowledge of the quality factors and sub-factors that influence the components in order to manage their quality effectively. However, previous component quality models haven't made an effort to systematically capture the most crucial quality elements that may affect the quality of the component at any point in its development. Since there is no overall framework that details the relationship between quality indicators and their constituent elements, developers have no way to improve the finished product's quality. Establishing a procedure to assess the component's quality throughout construction is vital for fixing the issues with the component. The majority of studies examining software quality have been conducted on commercially accessible programs. Although it is understood that quality features are crucial, the connections between these traits and software metrics have not been investigated extensively until recently.

REFERENCES

[1] Ahmed, Salman. (2018). A Systematic Literature Review of Success Factors and Barriers of Agile Software Development. International Journal of Advanced Computer Science and Applications. 9. 10.14569/IJACSA.2018.090339.

[2] Asghar, Ikram & Usman, Muhammad. (2013). Motivational and De-motivational Factors for Software Engineers: An Empirical Investigation. Proceedings - 11th International Conference on Frontiers of Information Technology, FIT 2013. 66-71. 10.1109/FIT.2013.20.

[3] Martin DE LAAT **"**Design and validation of a Software Requirements Specification evaluation checklist"2019

[4] Khan, Hameed & Asghar, Ikram & Ghayyur, Shahbaz & Raza, Mohsin. (2015). An Empirical Study of Software Requirements Verification and Validation Techniques along their Mitigation Strategies. 2321-5658.

[5] Ghayyur, Shahbaz & Ahmed, Salman & Ullah, Saeed & Ahmed, Waqar. (2018). The Impact of Motivator and Demotivator Factors on Agile Software Development. International Journal of Advanced Computer Science and Applications. 9. 10.14569/IJACSA.2018.090712.

[6] Anurag Dixit et.al "Umbrella: A New Component-Based Software Development Model" IPCSIT vol.2 (2011) © (2011) IACSIT Press, Singapore

[7] Hasan Kahtan et.al "2DCBS: A Model for Developing Dependable Component-Based Software" doi: 10.20944/preprints201608.0155.v1

[8] Amandeep Bakshi, "Component Based Development in Software Engineering" ISSN: 2277-3878,Volume-2 Issue-1, March 2013

[9] Thomas Murphy et.al "An analysis of non-observance of best practice in a software measurement program" doi: 10.1016/j.protcy.2012.09.006

[10] Mr. Sandeep Chopra et.al **"**Comparative Study of Different Models in Component Based Software Engineering" International Journal on Emerging Technologies (Special Issue NCETST-2017) 8(1): 441-445(2017) (Published by Research Trend, Website: www.researchtrend.net)

[11] Umesh Kumar Tiwari, et.al "Component-Based Software Engineering" https://doi.org/10.1201/9780429331749

[12] Andrzej Beniamin BUJOK **"**Development of a Software Testing Best Practice Framework for Medical Device Software"2020

[13] Sharanjit Kaur et.al **"**A Review on Automatic Detection of Dental Tooth Decay in Bitewing Radiography" Volume: 05 Issue: 03 | Mar-2018

[14] Tomar, Pradeep & Gill, Nasib. (2010). Verification & Validation of components with new X Component-Based Model. 10.1109/ICSTE.2010.5608788.

[15] Afrah Umran Alrubaee et.al "A Process Model for Component-Based Model-Driven Software Development" Information 2020, 11, 302; doi:10.3390/info11060302