International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 1 | ISSN : 2456-3307

# Study of Factors Affecting Reliability in Software Development Process

**Dilip Sadhankar[1], Ashish Sasankar[2]**

[1]Research Scholar, Department of Electronics & Computer Science, R.T.M. Nagpur University, Nagpur, Maharashtra, India
[2]Assocate Professor, Department of Computer Science, G.H. Raisoni Institute of Information Technology, Nagpur, Maharashtra, India

## ABSTRACT

A timely delivered software product only if all the phases of software development process are completed within estimated and mostly set up time. A software product with higher reliability, higher performance and functionality is the demand of fast changing market demands. Many researchers have made significant tools and techniques to accomplish the quality of software. However, at the same time, the field have need of a future research work to improve the quality of software and to cut the challenges in each phase. Software development processes issues have been in and around ever since the beginning of software development. Software system development is viewed as a series of discrete ordered activities that produce successively more constrained models of the system by binding in additional system aspects. Treating the system aspects as separate concerns allows software engineering techniques that control production cost and enhance reliability to be applied to each step. The greatest gains, however, are due to the reliability and traceability of the system over its lifetime. In this paper, we present a comprehensive overview of productivity factors recently considered by software practitioners. This paper also describes the major activities in the software development along with its key issues.

**Keywords :** Effort, Function Point, Reliability, Empirical Testing, Factors, Development

## I. INTRODUCTION

During the past few decades, software development process has been the centre of attraction for many software engineering researchers. Despite of the details of products and customers, software is an area where almost every industry is involved [1]. A successful software development process is therefore vital for industries in various fields. A systematic move towards the design, development, testing, and maintenance of software [2]. Figure 1.1 shows the different phases involved in the Software Development Lifecycle (SDLC). The main phases in the software development process are:

- Planning
- Requirements Analysis
- Design
- Implementation
- Testing
- Maintenance

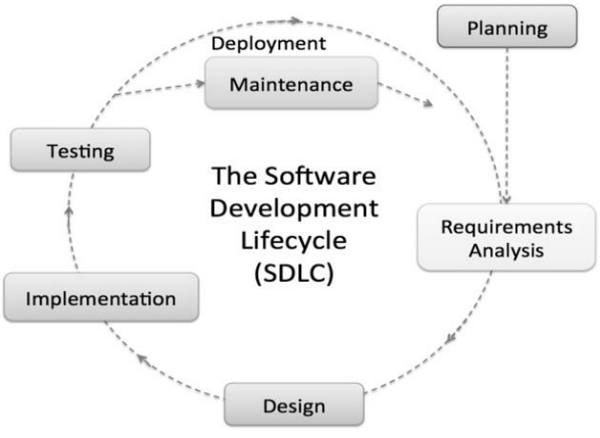


Figure 1: phases of SDLC

**Source:** Hans-Petter Halvorsen, 2017

In each phase, there are number of factors which differentiate the software development processes and

results in different quality levels of the final software product.

Software development, being a human-centered process, human factors has a great impact on the process and its performance. Human factors in this process can be examined or studied from different perspectives such as psychological, cognitive, management and technical aspects. Beside this, human factors have different levels of impact in the process varying from organizational and interpersonal to individual. Even though human factors have been proving to have impact on software development process, unfortunately the researchers in the software engineering and development research areas have overlooked them. Thus, there seem to be a need to recognize and characterize human factors and their impact on development process. A systematic review over software development human factors could draw attention to the research needs and as a result improve the research in the Software engineering field.

The rest of this paper discuss the scope and need of software engineering, diagnosing development performance and issues affecting software development.

## II. BACKGROUND

Software and software systems are getting more and more complex, so it is important to have the necessary "tools" in your "toolbox" to be able to create and maintain software. Software Development is a complex process, and it may involve a lot of money and a lot of people. Here are some examples:

- Windows 7: A Team with 1000 Developers created Windows 7
- Number of Code Lines: Real systems may have millions of code lines
- Big money: 100+ million Development Projects
- Combination of Hardware and Software: Most of the projects involves both hardware and software and integration between them.
- iPhone 1: Development period 2004-2007, 1000 Apple employees worked with the device, Estimated cost: $150 millions.

Project Planning and Management is important in Software Development and it uses different approaches to deal with the Software Development.

## III. SCOPE AND NECESSITY OF SOFTWARE ENGINEERING

Software engineering is an engineering approach for software development. Developers alternatively view it as a systematic collection of past experience. The experience is arranged in the form of methodologies and guidelines. A small program can be written without using software engineering philosophy. But if one wants to develop a large software product, then software engineering principles are crucial to achieve a high-quality software cost effectively [3].

Without using software engineering principles it would be difficult to develop large programs. In industry it is usually needed to develop large programs to accommodate multiple functions [4]. A problem with developing such large commercial programs is that the complexity and difficulty levels of the programs increase exponentially with their sizes as shown in figure 1.2 . For example, a program of size 1,000 lines of code has some complexity. But a program with 10,000 LOC is not just 10 times more difficult to develop, but may as well turn-out to be 100 times more difficult unless software engineering principles are used [3]. In such circumstances software engineering techniques come to salvage. Software engineering helps to reduce the programming complexity. Software engineering principles use two important techniques to reduce problem complexity: abstraction and decomposition [5, 6].

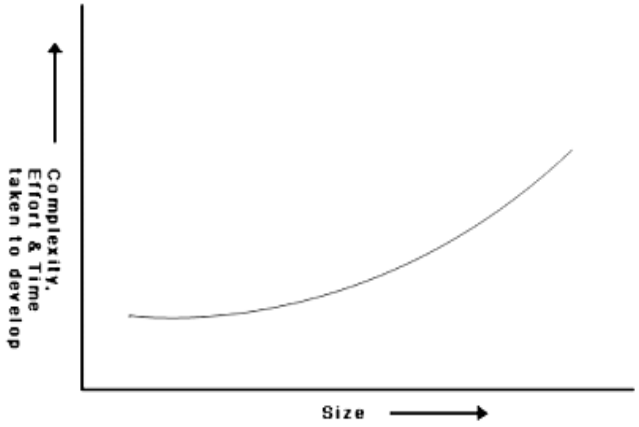


Figure 1.2 Increase in development time and effort with problem size

## IV. LITERATURE SURVEY

***Adam Trendowicz, Jurgen Munch (2009)*** studied factors influencing Software Development Productivity. In addition, instead of factors influencing development productivity, some studies considered a specific factor's value and its specific impact on productivity. For instance, some studies identified the life cycle model applied to developing software as having an impact of productivity, while others directly pointed out incremental development as increasing overall development productivity.

***Petra C. De Weerd-Nederhofl (2001)***, shares his experience on doing his research project on organizing and managing new product development systems. In a similar work by Ganesh N Prabhu (2005), seeks to understand the complexities of the new product development process with a panel discussion at IIM-Bangalore, India. The panel reviewed the emerging opportunities in new product development in India, strategies for the development of the new products, product development capabilities, and issues and challenges in the commercialization of new products.

***Walket Royce (2002***), stated on some of the techniques involved with reducing the size or complexity of the software and improving the software development process. The main thrust of process improvements is to improve the results of the productive activities and minimize the impact of overhead activities on personnel and schedule.

***Magne Jorgensen (2004***), in the experiment conducted indicates that customer expectations have surprisingly large impact on software development effort estimates, even when the estimators are told to disregard this information.

**Peterson** has discussed the use of systematic mapping, review and their procedure in software engineering related areas. While **Michael Unterkalmsteiner** have addressed evaluation and measurement of Software Process Improvement by reviewing 148 papers published between 1991 and 2008, **Magne Jørgensen** and **Martin Shepperd** carried out a systematic review on Cost Estimation by reviewing 304 papers published in 76 journals. **Tore Dyba** systematically identified and analyzed all the existing studies on pair programming. **Tracy Hall** reviewed studies of motivation in software engineering published between 1980 and 2006. **Laleh Pirzadeh** reviewed 92 studies of motivation in software engineering that were published in the literature between 1980 and 2006. Moreover the databases used for review were different from the review (including ACM Digital library, EI Compendex, Google Scholar, IEEE Explore, Journal of Systems and Software, etc). Additionally review had different inclusion and exclusion criteria such as including conference proceedings, or conference experience report).

Research in the software development process and factors that impact the process has been conducted. Sawyer and Guinan (1998) presented their studies that described effects on software development performance due to production methods of software development and the social processes of how software developers work together need to be considered.

Roberts Jr. et al. (1998) identified five factors important to implementing a system development

methodology (SDM). Realizing the importance of factors that influence the software metrics.

Japanese researchers **Furuyama** (1994, 1997) studied factors such as working stress, development methodologies, etc. using design of experiment methods [18].

## V. DIAGNOSING DEVELOPMENT PERFORMANCE

Software development is a very complex process which engages human beings, underlying systems, nature of applications and so on. It is a dynamic process that differs from project to project. In our modern society, software has become a very important component in all kinds of systems and software failure has become the most crucial factor that terminates the service and proper function of the whole system. Therefore, it is very important and urgent to realize the software development process and eliminate as many potential problems in software as possible [11].

Companies aspiring towards better software development must first understand their starting points. Beginning any software development improvement idea with a complete diagnosis give companies the advantage of rooting their improvement plan in a deep understanding of their existing positions.

Ability to attentively address the questions that underline the fundamental drivers of the software development success is there behind the performance of the top companies [20].

### What software is being developed?

Companies need to review and prioritize different feature requirements, scope and need to manage requirements (including late requirements.). They should also assess and understand how they set up the software architecture and system design to drive efficiency, for example, exploit code reuse and ensuring a modular software architecture with clear interfaces and interdependencies [11].

How the software developed?

At the beginning, project planning and efficient resource management need to be evaluated. Companies must also review their existing software methodology and start to see if there are new and superior processes available to increase productivity, quality and time to market, for example, by moving away from traditional waterfall methods to agile software teams and development teams with integrated operations expertise [11].

Where is software developed?

The actual location of the development is also important. Companies must look specially at their decision to oursource versus develop internally and the inner workings focused on software development. software development mediocrity requires careful analysis. The path towards software development improvement is highly customized effort, as no two organizations require the exact same approach [10].

## VI.FACTORS INFLUENCING SOFTWARE DEVELOPMENT PROCESS

A substantial amount of work was carried out to study the most important factors affecting the reliability of software development process by proposing and analysing processes, methods, tools, and best practices.

There are reviews on reliability factors in software engineering available in the literature. These works focus on the main dimensions of the product, personnel, project, and process. Each of these dimensions is then characterized by sub-factors: product is related to a specific characterization of software, such as domain, requirements, architecture, code, documentation, interface, size, etc. Personnel factors involve team member capabilities, experience,

and motivation. Project factors encompass management aspects, resource constraints, schedule, team communication, staff turnover, etc. Process factors include software methods, tools, customer participation, software lifecycle, and reuse [13].

Furthermore, any software development effort, even in the presence of skilled individuals, is likely to be unsuccessful if it does not explicitly account for how people work in concert.

A software development environment is a complex social structure that may dissipate the positive impact of skilful persons as well as software tools and methods if team communication and coordination will be unsuccessful. Factors makes possible team communication and work coordination are particularly vital in the context of software outsourcing. Geographical and, often, mental distance connecting the concerned parties require dedicated decision-making activities and communication services to maintain a acceptable level of productivity.

For the reason, tools and methods are considered as human support that amplifies the positive impact of highly skilled and well-coordinated teams on development efficiency.

Another most commonly chosen factors prop up the theory that schedule is not a simple derivative of project effort.

Several further factors are considered as either contributing to the quality and volatility of requirements or moderating the impact of already instable requirements. Distribution of the project effort concentrate on the requirements phase as well as valuable customer participation in the early phases of the development process are the factors most commonly thought to increase requirements quality and stability. Disciplined requirements management as well as early reviews and inspections on the other hand, moderate the impact of already instable requirements. . It seems that the first years of enthusiasm also brought much disappointment.

## VII. ISSUES AFFECTING SOFTWARE DEVELOPMENT

The successful development and delivery of products and services points towards company's ability to effectively develop quality software. However before issues can be tackled, development teams need to know what issues could arise in the development process. The software development process is complicated, Proper communication, planning and testing can help ensure that teams don't fall victim to the above problems [19].

- **The standards are out there but they're not enforced:**

There's a rapid transition in IT with all of the emerging technologies for emerging companies. Enterprise companies cannot embrace all the new technology because it's risky. It can be complicated to integrate programming languages and operations consolidating the new with the old. It is a challenge ensures users are secure with so many assets interacts with multiple applications and platforms. Domain knowledge is necessary. Everyone needs to be careful about keeping apps up to date and ensuring they are secure. Compatibility, security, and assets are challenges.

- **Not having a planned process for the entire SDLC increases waiting and queuing time:**

Without processes or enforcement quality is an issue. Lack of visibility and knowledge lead to uncertainty. Everyone needs visibility to acquire the knowledge necessary to work together. Collaboration across multiple teams and organizations requires the right mindset. Development team must know the objectives. While company wants to use latest technologies then company have to know the end

goal and determine whether or not the latest technologies are the best option.

- **Time to value is the biggest challenge:**

Automate everything to imitate every possible situation: pre-built model, automated build and tear down, more complexity with more customization.

- **Scope the building design solution and changes in architectural patterns**:

There is a lot of stir up throughout application development. There are people looking for newer ways to develop apps.

- **Lack of business value and planning:**

Companies must have certain attributes before becoming a great software company. Team needs a learning culture that's willing to make mistakes and perform empirical testing.

- **How testing is done:**

Make it easier to automate the testing cycle to move the software through the pipeline more quickly. Everything needs to be written in a way it can be automated. Work with customers on end-to-end processes so they can optimize. Turn an end-to-end view into an automated process.

- **Many companies are still not aware of security issues:**

There are three types of customers:

1) Information security officers who are concerned about data breaches;
2) CIO's who is concerned about availability and uptime, DDOS attacks and ransomware;
3) CFO's are concerned about fraud.

Development team typically works with the information security officers to address all three. There are so many backed-up requirements that all needs have been prominent to critical. So much of everything is software overwhelmed with a throughput of needs. There are not enough people or companies in technology to meet the needs and this is resulting in a log jam around the world.

- **Cultural – DevOps struggles:**

Need to change the state of mind from the scratch to blame developers responsible for a stable environment.

- **Complexity:**

Developers are bombarded with too many options, frameworks to solve the problem. Companies are trying to make the lives of software developers easier – customization is a small part of a complex system. People don't know how to optimize the code in all of the different environments. There is need to be able to see what is going on in the code and the frameworks to ensure the application is written and performing as efficiently as possible.

- **Not having the right processes in place.**

Doing Agile wrong, not doing sprints, measuring velocity, being iterative. Have processes in place to make it happen. It's a process with greater rigor.

## VIII. CONCLUSION

Software development process is very much technical. There is a need of proper coordination among the team members to complete the product development in time. Development is an in-house process most of the time and customer satisfaction is an outdoor process to be managed effectively.

This paper has presented a wide-ranging overview of the literature and study regarding the common factors influencing software development. The major outcome of the study is that the success of software projects still relies upon humans. The second most usually considered factors are tool and method. Yet, tool or method alone are not best factors and cannot be replaced for highly skilled people and effective work coordination. Still, Investing in people is considered more beneficial than investing in tools

and methods only. The excess of factors that should be considered to gain the probable benefits from reuse might enlighten this situation. However, it must be considered that the analyzed studies typically vary widely with respect to the acknowledged factors, their interdependencies, and their impact on productivity. Therefore, every organization should think of potential productivity factors in its own environment ("what is good for them does not have to necessarily be good for me"). In addition, since software is recognized as a very rapidly changing environment, selected factors should be reviewed and updated regularly. The particular project data must be collected, analyzed, and interpreted from the perspective of the stated productivity objectives. Incompatible measurements and/or derisory analysis methods may lead to misleading conclusions about productivity and its influencing factors. One may say that precise measurement processes and adequate analysis methods have a vital impact on productivity, although indirectly. Therefore, such aspects as clear definition and quantification of selected factors, discovery of factor interdependencies, as well as quantification of their impact on productivity has to be measured.

## IX. REFERENCES


[1] Laleh Pirzadeh, "Human Factors in Software Development: A Systematic Literature Review" Goteborg, September 2010

[2] Software development – A practical Approach, book, Hans-Petter Halvorsen, ISBN 978-82-691106-0-9

[3] "Basic Issues in Software Engineering Version 2" CSE IIT, Kharagpur.

[4] I. Sommerville, Software Engineering: Pearson, 2010.

[5] E. J. Braude and M. E. Bernstein, Software Engineering. Modern Approaches, 2 ed.: Wiley, 2011.

[6] F. Tsui, O. Karam, and B. Bernal, Essentials of Software Engineering, 3 ed.: Jones & Barlett Learning, 2014.

[7] Dr.S.K.Nagarajan, Dr. Vanathi Vembar and K.Anandhan, 2011, "Managerial issues in software product development", 3rd International Conference on Information and Financial Engineering IPEDR vol.12 , IACSIT Press, Singapore

[8] Petra C. De Weerd-Nederhofl., "Qualitative case study research. The case of a PhD research project on organizing and managing new product development systems", Management Decision, 2001, 39/7, pp 513-538.

[9] Walket Royce., "Improving software development economics : Reducing Software product complexity and improving software processes", Information Technology, 2002. May, pp 53-57.

[10] Kai Petersen, Robert Feldt, Shahid Mujtaba and Michael Mattsson, 2008, "Systematic Mapping Studies in Software Engineering", EASE'08 Proceedings of the 12th international conference on Evaluation and Assessment in Software Engineering

[11] Software development handbook, 2016

[12] B.W. Boehm, "Improving software productivity,"

[13] S. Wagner and M. Ruhe, "A structured review of productivity factors in software development," Institut für Informatik, Technische Universität München, techreport TUMI0832, 2008.

[14] K.D. Maxwell and P. Forselius, "Benchmarking software-development productivity," IEEE Software, Vol. 17, No. 1, Jan. 2000, pp. 80–88.

[15] M. He, M. Li, Q. Wang, Y. Yang, and K. Ye, "An investigation of software development productivity in China," in International Conference on Software Process. Springer, 2008, pp. 381–394.

[16] Luigi Lavazza, "An Empirical Study on the Factors Affecting Software Development

Productivity", e-Informatica Software Engineering Journal, Volume 12, Issue 1, 2018, pages: 27–49, DOI 10.5277/e-Inf180102

[17] Zhizhong Jiang and Peter Naude, "An Examination of the Factors Influencing Software Development Effort", World Academy of Science, Engineering and Technology International Journal of Computer and Information Engineering Vol:1, No:4, 2007

[18] Tsuneo Furuyama, "Analysis of Factors that Affect Productivity of Enterprise Software Projects",

[19] Tom smith, "Issues Affecting Software Development Today", https://dzone.com/articles/issues-affecting-software-development-today

[20] Paul Clarke, 2012 "The situational factors that affect the software development process: Towards a comprehensive reference framework", Journal of Information Software and Technology, Vol. 54, Issue 5, May 2012. pp. 433-447