ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 9(27), DOI: 10.17485/ijst/2016/v9i27/97591, July 2016* ISSN (Online) : 0974-5645

The Use of Empirical Software Engineering to Solve the Software Crisis

Rahim Lotfi* **and Ahmadreza Ghasemian Dastjerdi**

Department of Software Engineering, University of Isfahan, Isfahan, Iran;
Mr_lotfi@ymail.com, Ahmadrezaghasemian@yahoo.com

Abstract

Objectives: Like other sciences, software engineering needs to use physical models and follow empirical patterns in order to improve the software development cycle. **Methods:** The experiments conducted in the real world produce more acceptable (realistic) results than do methods such as surveys or interviews. Most software development projects, whether small or large, do not end in success. **Results:** This means that the software crisis has still remained unsolved and some software engineers have wrongly regarded the incorrect selection of methodology as the main cause of this problem and instead of finding a solution to the problem, they seek a proper methodology that can complete all software development projects successfully. **Conclusion:** We deal in this article first with the reasons for the success and failure of software projects and finally suggest a method to transfer experience.

Keywords: Empirical Software Engineering, Methodology

1. Introduction

There are many articles on empirical work in software engineering, some of which cover new ideas and information and are suitable for publication. Like empirical disciplines such as physics, medicine, architecture and many other disciplines, Empirical Software Engineering empirical is a good way for software development based on empirical components. Software engineering requires high – level methods for modeling and testing, and cannot reach the production cycle only through observation reliance on logical thinking[1]. Software engineers can manage and finish a project successfully using the right methodology and the use of the power of experience. Methodology is only a framework for managing a software process production and cannot have a significant role in the success of a project by itself. Success in a project is like the flight of a bird requiring two wings (methodology and experience). In other words, methodology and experience is each a necessary, but insufficient condition for the successful implementation of a project. On this basis, we should determine using techniques and experiments what tool works in a specific time interval and how it works, and should also design using the experience that we have and considering the limitations and the performance of the products for improvement of the project process and better learning. The purpose of empirical software engineering is to use individuals' experiences in the software management, development and improvement, and methodology is an algorithm to organize this process. Methodology is used as sustainable activities to help the development activities and exists in all phases of the software production process and uses different symbols for the expression of different purposes and the production process at a heterogeneous level.

2. Why Do We Need Empirical Studies in Software Engineering?

Empirical studies provide us with the systematic and measurable principles as well as the ways to control human-based activities. Experience is needed to help make the lab environment for both researchers and

* *Author for correspondence*

developers. For example, the study of empirical work has shown that software development based on modeling is more useful and efficient than other methods. Sometimes the results of an empirical study may not be so strong, but it may have new and innovative approaches.

2.1 The Accuracy Rate of Empirical Sciences

We do not want to re-produce the products. Rather, our purpose is to use the experience gained from previous products in order to improve future (new) products (new). The use of individuals' experience is an art that does not follow the pattern of research and experience can be used to help better understand the effects of the use of special tools in some environments. Although none of the empirical sciences is absolute and flawless, sometimes a comparison between empirical work and the standards of texts rejects many interesting and significant results.

The accuracy of empirical sciences depends on the type and method of experiment (interviews, observations, and survey) and how often it is repeated. The more an experiment is repeated, the more reliable the accurate the obtained data will be. Figure 1 shows the relationship between experiment repetition and the accuracy of the results obtained.

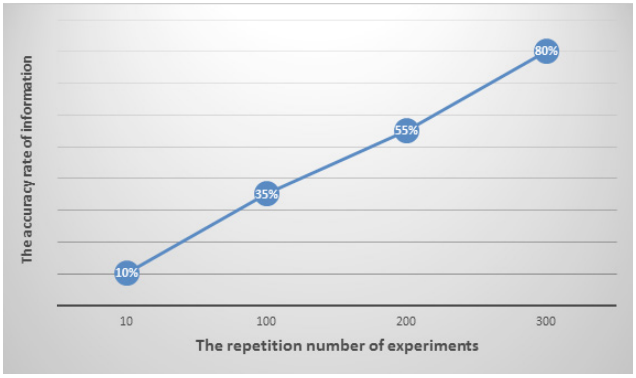

Figure 1. Relationship between repetition of experiments and the results' accuracy.

3. Goals and Benefits of Empirical Studies in Software Engineering

The aim of empirical studies in software engineering is to use individuals' experience to improve the management of the software development cycle. One of the important benefits of empirical studies in software engineering is to raise the abilities of the development team to solve the problems over time. Some of the experiments or empirical studies are conducted in order to reject or confirm a theory or get access to new and relevant results and they can be categorized in the following way.

- Exploration (to discover new things)
- To look for new insights
- To try to find answers to questions
- To measure (the products quantitatively and qualitatively)
- To describe (the detailed features of the circumstances, events, projects)
- To explain (the problems in a scientific way and the relationship between them)
- To find the relations
- To find the differences and changes

The empirical studies and researches performed in software engineering to improve software development and its main objectives can generally be classified in the following two categories:

- Leveling (aligning) the academic researches with science and industry.
- Increasing the knowledge (knowledge of tools and products).

One can add experiences to his working device (software engineering in practice) by evaluating the studies and scrutinizing the theories and models. Figure 2 shows two basic concepts. The first concept is the use of the concept of theories in the science projects of software engineering and the second one is the alignment of academic research with industrial applications.

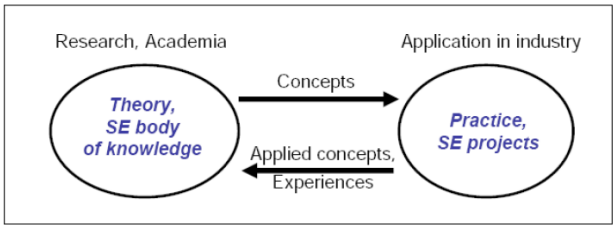

Figure 2. How to apply the results of researches in practice[2].

4. The Effect of the Selection of Methodology and Experience to the Success of Software

Methodology is not merely a framework for managing the production process of software. Therefore, it cannot have a significant role in implementation of the software development process. Of course, this does not suggest that all methodologies are the same and follow the same process. For example, to explain this point more, if a company with a methodology such as Incremental Methodology finishes its project successfully, it can definitely succeed to some extent with methodologies such as Spiral Methodology, or it may happen that a company with Incremental Methodology becomes successful, whereas another company is unsuccessful with the same methodology. This is not say that the selection of methodology plays no part in the success of a project, but another more important factor that can play a significant role in the management and control of the development process of a software is a experience and how to assess the risks and how to respond them, since the agile development of a software is a very dynamic process which we cannot do within a static framework. That's why the agile methodologies are more successful than heavyweight ones, as the former methodologies are more flexible and dynamic. By considering the characteristics of agile methods and their relationship with empirical techniques in large and small teams, we will notice the high correlation between them. Experts and professionals got together to improve the software development methods and they brought about agile methodologies by collecting a set of experiences and using them. In other words, an agile methodology is a set of rules and procedures recommended by experienced professionals[3]. Therefore, agile methodologies are based on experience, and experience is indeed the heart of such methodologies[4], so that they are actually based on experience and conditions instead of being rule-based and acting according to the program. The use of experience if important for success in agile methodologies, but production is more important than that. Agile methodologies have been made based on "feedback" and "changes"[5]. However, agile methodologies require quick (instant) access to the knowledge base of experiences and experiences exist in the knowledge base as knowledge. Using and testing these experiences helps remove the shortcomings from the previous experiences and leads to the creation of new experiences. Thus, agile methodologies provide a useful environment for empirical research, and as shown by studies, these methodologies are inexpensive and efficient workshops to show empirical knowledge. This does not mean that we do not need heavyweight methodologies; rather, it is obvious that it is inevitable to use heavyweight methodologies in large projects, or methodology is not at all needed in such projects, so that it can be said that a methodology is needed for any individual or project. This sentence implies that no single methodology can be defined for individuals and projects in a fixed way, because projects and the members of the software development team require their own special conditions. However, some engineers wrongly select a single methodology and follow the process of that methodology until its stages are completed. In fact, they get along with the steps of methodology instead of keeping up with the process of the project, and thus fail in most cases and lead to a software development crisis. This theory does not mean that software engineers do not need to know methodology or to ignore their steps, but understanding various methodologies (Rup, Agile, Incremental .....) brings engineers an open view and insight to manage the project and even producing different versions of a single software may require its own specific methodology and conditions. This is because software development is a completely dynamic process which cannot be limited to a specific methodology. As methodology earlier, methodology is completed with experience. Methodology is like the skeleton while experience is its soul. There are many people and companies that use methodology (Y) and succeed in their projects. Still, there are individuals and companies that use the same methodology (Y), but fail in their projects. This statement indicates that methodology required something to be completed, and that something is experience. Methodology selection is important, but not as important as a software development. This paper presents a new approach to solve this problem.

5. Suggesting a Solution

Soft engineering software has been suffering from the software crisis for many years and engineers and specialists

have not succeeded in presenting a good answer to the problem. Below we propose a method, hoping that it proves useful and is completed and implemented with engineers' help. For each project, whether large or small and successful or failed, we have to prepare a manual called "biogeography of Project x" or "production cycle of Project x". This manual must contain the complete stages of the project, telling like a story or tale how to attract customers, the first day and the end of the project when it will be completed.

The manual includes:

- How to attract customers
- How customers should request
- The overall process of the project (the story-like biogeography of the project)
- The factors (individuals, organizations, tools, ...) influencing the project.

At the end of the manual, the members of the software development team, the beneficiaries, customers, or any organization which has influenced the process of the project have been asked to write their views, role and effect on the project with their own language and wording in the three to five pages left blank for this purpose.

What should be considered in the preparation of the final part (poll) of the manual is the freedom of speech and honesty on the part of those that have influenced the project, so that all can write what they think with their own language. More importantly, we should ensure them that no reform or correction will be made in their views and they can be published as an eBook or manual.

6. The Benefits of the Suggestion

The benefits of this suggestion are clear. This suggestion provides a way to transfer experience. The suggestion has the following benefits:

- Software engineers can get the required experiences and apply them in similar conditions by reading a few biographies of software development and reviewing the weaknesses, mistakes, achievements, and the right and influential decisions.
- Reading this manual can help the reader evaluate the success rate of a person.
- It helps managers to identify and recruit the qualified personnel and workforce more easily.

7. Conclusion

We first dealt with the definition of empirical software engineering and its impact on the project process and management, explaining why we need empirical software engineering and discussing the accuracy rate of the results and the empirical data. In the next step, we evaluated the effectiveness of the methodology and experience in the success of a software, and we finally ended the discussion with suggesting a method to transfer experience.

8. References

1. Basili VR. The role of experimentation in software engineering: past, current, and future. Proceedings of the 18th International Conference on Software Engineering. IEEE Computer Society; 1996.
2. Boehm B, Hans DR, Zelkowitz MV. Foundations of empirical software engineering: The legacy of Victor R. Basili. Springer Science & Business Media; 2005.
3. Javid MA, Ghomashi H. Thermodynamics analysis and optimization of abadan combined cycle power plant. Indian Journal of Science and Technology. 2016; 9(7):45–70. DOI: 10.17485/ijst/2016/v9i7/87770.
4. Williams L, Alistair C. Guest editors' introduction: Agile software development: It? S about Feedback and Change. Computer. 2003; 36(6):39–43.
5. Dyba T, Torgeir D. Empirical studies of agile software development: A systematic review. Information and Software Technology. 2008, 50(9–10):833–59.