International Journal of Research in Advent Technology (IJRAT)
(E-ISSN: 2321-9637)
Special Issue National Conference "CONVERGENCE 2018", 09th April 2018

Software Engineering Research Framework: Extending Design Science Research

Pooja Wele[1], Dr. Pradip Jawandhiya[2],

Department of Computer Sci. & Engineering

Pankaj Laddhad Institute of Technology and Management Studies, Buldhana

Email: welepooja@gmail.com

ABSTRACT-Software Engineering may be regarded as a relatively young discipline, which has been driven by technical innovations, trends and practices. Although a great many research studies explored solutions, fundamental problems in SE domain still exist. Most of the SE research efforts adopt the principles of quantitative, qualitative paradigms, or both, there are also models, such as Design Science Research (DSR), which can address some of the issues in SE research domain. However, there is a need for a research model, which considers the prescriptions of research paradigms as well as the theoretical and trans-disciplinary foundations of SE as an applied discipline. In this paper, therefore, we propose a research framework by extending the DSR method and leave its empirical and theoretical support to the future studies. Our effort may be considered as a preliminary attempt rather than a complete solution

Index Terms- Software engineering, Software engineering sub disiplines, Research method, Design science research.

1) INTRODUCTION-

When compared to other academic disciplines from an evolutionary point of view, Software Engineering (SE) may be regarded as a relatively young discipline. Technical innovations changing from time to time have been a major driving force for SE trends and practices. It has been usually driven by industrial needs, and thus, languagecentered computer programming has been dominant in SE [1]. However, the fundamental problems in SE still exist today [2]. Therefore, a wide range of topics, such as programming paradigms, methods,

quantitative research methods

DSR would be suitable and applicable to SE, however, our claim is that it may be extended. The rationale behind this assertion is based on two reasons. First, the need for the elaboration and clear indication of the role that a theory and its constructs play at a SE research. The investigation of the theory uses in SE experiments reveals that a quarter of the models, tools and techniques, which are also combined with people and technology, have been in the research interest of academicians and practitioners.

In this context, some of the studies specifically focus on the research methods that aim to frame and explore theories and knowledge base in SE. These studies are based on

studies use theory for only explanatory or motivational purposes, and it indicates no evidence of theory-driven SE research [9]. Second is a process model that can integrate the prescriptions of other research paradigms while regarding the unique characteristics of SE as an applied discipline. Of course, describing the possible research methods in detail and how to adopt them for SE is far beyond the scope of our study. Thus, we only attempt to offer a research framework inspired from DSR, and leave its empirical and

theoretical support to future studies. The rest of this paper is organized as follows: Section 1 introduces the sub disciplines. Section 2 introduces the DSR. Section 3 presents the proposed research framework, and finally, the paper is concluded in Section 4.

2) SUB DISIPLINES OF SOFTWARE ENGINEERING-

2.1 Software requirements:

The elicitation, analysis, specification, and validation of requirements of software.

2.2 Software design:

The process of defining the architecture, components, interfaces, and other characteristics of a system or component. It is also defined as the result of that process.

2.3 Software component:

It is also detailed creation of working, meaningful software through a combination of coding, verification, unit testing, integration testing, and debugging.

2.4 Software testing:

The dynamic verification of the behavior of a program on a finite set of test cases, suitably selected from the usually infinite executions domain.

2.5 Software maintenance:

The totality of activities required to provide cost effective support to software.

2.6 software engineering process:

The definition, implementation, assessment, measurement, management, change, and improvement of the software life cycle process itself.

3) DESIGN SCIENCE RESEARCH-

Design activities exist in many fields of applied disciplines, such as engineering, architecture, education and fine arts, and thus, research in design has a long history. However, design science approach is mainly a problem-solving paradigm seeking to create innovative artifacts, ideas, and products through analysis, design, and implementation processes [10]. It is claimed that a DSR should have theoretical background as well as its practical implications, justified theory and utility are valued equally. Being directly relevant to IS and SE, the DSR paradigm addresses these two issues, and puts a priority on relevance at the application domain of IS. On the other hand, professional design is the application of existing knowledge to the problems using state-art-technology, methods or best practices that again exist in the knowledge base. However, DSR focuses on solving new or known problems using unique or innovative techniques for contribution to the current

knowledge base. This is a cyclic process, and thus, as the results of a DSR is approved and codified in the knowledge base, however, it becomes routine applications to the known problems.

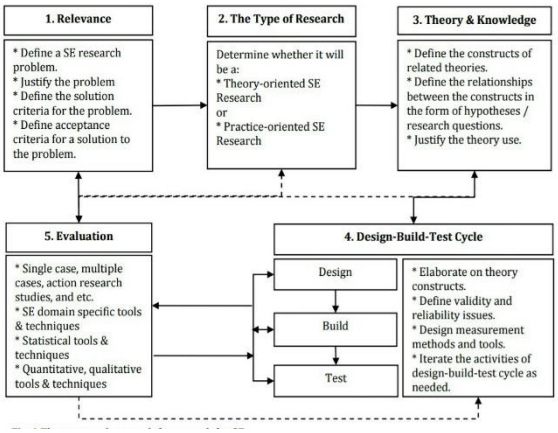
3.1 **Indicate the guidelines for a DSR as follows:**

- The research objective should be to develop technology-based solutions to the relevant and important business problems.
- It must produce a practical artifact, which would be in the form of a method, a model, a construct or an instantiation.
- The utility, efficacy, and quality of a design artifact must be rigorously demonstrated with valid and reliable methods and tools.
- An effective DSR should provide clear and verifiable contributions especially in the research domains of the design artifact, design foundations and methodologies.
- A DSR must be effectively presented to the related audiences of both technology and management.

4) THE PROPOSED FRAMEWORK FORSOFTWARE ENGINEERING RESEARCH-

Our proposed research framework for SE is presented in Figure 1:

fig.1 The proposed research frame work for Software Engineering



4.1 **Relevance**

A good DSR on SE usually begins with identifying the problems or representing opportunities in an '11111111111actual SE application environment. While the ultimate SE artifact is expected to improve the knowledge base, the output from a research must also be returned into this environment. Therefore, the relevance base provides the requirements for the research and defines the acceptance criteria for the evaluation of both research results and software products. When defining a relevant research problem in SE domain, all stakeholders of a software project have to justify the problem, and also define the solution and acceptance criteria for that problem.

4.2 **The Type of Research**

Empirical SE research may have different research methods [14] with four main phases, which are definition, planning, operation, and

interpretation respectively. An important decision that a researcher must make is to determine the general research objective as such whether the study will be practice-oriented or theory-oriented research. Because, the decision will also determine (a) the way in which the exploration is conducted, (b) the arrangement of cases and selection of instances for the study, (c) and the implications of the study's outcomes [11].

A practice is the real life SE situation for which a practitioner has a responsibility, and in which (s)he may act or must act. A practice-oriented SE research is mainly aimed at contributing to the knowledge of specific organizations, practitioners responsible for a particular practice. In this type of research, the stakeholders are interested in knowing whether the treatment or intervention makes a positive change in the concrete circumstances of that practice to which the SE research is oriented. Therefore, the success criteria for a practice oriented study are: (a) whether the specific outcome(s) has been achieved, (b) the intervention(s) has made a positive difference in the circumstances of the practice, and (c) empirically correct conclusions about the study are reached [11].

On the other hand, the focus of a theory-oriented research may be establishing the correctness of a proposition(s) or theory development itself in a SE context while the ultimate outcomes may be useful for the practice. The empirical SE intervention would not benefit the organization, and thus, the purpose of a theory-oriented study is to contribute to the generalizability and robustness of theoretical explanations and predictions [11]. However, whatever the main concern of a SE research study would be, both of the research types require methodologically correct, systematic data collection, and evaluation of observable facts in a SE context.

4.3 Theory and Knowledge Base

SE requires both empirical and theoretical research. Empirical studies explore, predict and try to explain the investigated cause-effect relationships between constructs of a theory, and find out what types of SE constructs should be used in what situations and circumstances. A SE experiment is, therefore, the primary research method directly to make comparisons and observe the effects of measures taken to improve a SE process. Therefore, theories are commonly viewed as a coherent set of tested propositions, which are generally regarded as correct, and able to predict or explain facts or phenomena in SE. As having potential use to practitioners and researchers, a SE theory provides a conceptual framework for explaining observed phenomena as well as it helps understanding the basic concepts and underlying mechanisms of software systems and their behaviors [12].

It is suggested that the four main parts, such as (a) Constructs, (b) Propositions, (c) Explanations, and (d) Scope, comprise the structure of a SE theory [12]. When supporting SE research studies, a theory helps to develop and combine research efforts, and it facilitates communication of knowledge and ideas. As to the industry, it can provide software decision-makers with required input regarding the selection of a method, tool or technology for a software project. There may be three modes of

theory use in a SE research: (a) using theories from other disciplines as they are, (b) adapting theories generated in other disciplines to SE, and (c) generating theories from scratch in SE discipline [12].

Whether a study on a SE topic is on a theory or practice oriented, theories play an important role for the research area of SE [13]. Thus, the theory & knowledge base of our proposed framework provides scientific theories to a SE research and it includes SE methods, past and the state-of the-art knowledge in SE domain. As Hevner et al. [8] indicate, this phase ensures that the intended software designs or products are not routine applications based upon well-known software processes; rather, they are really research contributions to the SE discipline.

Our proposed research methodology shares the same principles of Gregor and Hevner's [4] framework for evaluating the knowledge and research contribution of a research study in SE discipline (Table I). Accordingly, if we apply known solution to the known problems, the result is a "routine design" with no contribution. For a known problem-new solution case, the result would be an "improvement" with knowledge contribution. The nontrivial extension of a known solution to a new problem results in "exaptation" with knowledge contribution.

Finally, application of a new solution to a new problem is an "invention" with an important contribution to the SE theory and knowledge base.

Table 1: Knowledge contribution, adopted from [4].

Problem Domain Maturity	**Solution Maturity**	**Research Knowledge Contribution**
Known problem (high maturity)	**Known solution (high Maturity)**	➢ **No contribution** ➢ **Routine design**
Known problem (high maturity)	**New solution (low Maturity)**	➢ **Improvement** ➢ **Research contribution**
New problem (low maturity)	**Extension of known solution (high maturity)**	➢ **Research contribution** ➢ **Exaptation**
New problem (low maturity)	**New solution (low maturity)**	➢ **Research contribution** ➢ **Invention**

The skillful selection and application of appropriate theories is also important for constructing and evaluating the software artifacts. In addition to the results of a DSR on SE, the contributions will include any extensions to the original theories, methods, and the experiences gained from performing the research and field testing. To form and justify the theoretical base, the researcher(s) should be able to define the constructs of the theories related to the SE research and the

relationships between the constructs either in the form of hypotheses or research questions.

4.4 Design-Build-Test Cycle

The main activities of this phase are:

- Elaborating on theory constructs, *+3
- Defining validity and reliability issues on the research,
- Designing measurement methods and tools,
- Iterating the activities of Design-Build-Test Cycle

The Design-Build-Test Cycle phase is where the most of the work is done and by which the critical research activities are conducted. The requirements obtained from the Relevance Phase (the solution and acceptance criteria) and also the guidelines of the previous phase (Theory & Knowledge Base), such as the software constructs and their relationships, form the baseline. As indicated, this phase is an iterative and incremental process, which includes (a) the generation and evaluation of design alternatives, (b) selecting one of them, (c) building the artifact, and finally (d) testing until a satisfactory solution is achieved [6]. It is important to keep a balance between the efforts spent in building, testing the evolving software artifact and the other research activities. It is highly probable that this phase will necessitate multiple iterations of the cycle, and therefore, the targeted software artifacts must be tested in laboratory and experimental environments before releasing the artifact into field testing at the Evaluation Phase. It is worth to note that this phase is the heart of a SE research project where the most of the contributions to the SE theory and knowledge domain are expected [4].

4.5 Evaluation

At this phase, the evaluation of the software artifact is done with single or multiple case studies [11] combined with the use of quantitative and/or qualitative methods as well as SE domain specific tools and techniques. The evaluation result will determine whether additional iterations of design-build-test cycle are needed depending on the requirements obtained from the Relevance Phase. Moreover, the feedback from Evaluation Phase and the results of field testing may also lead to the restatement of research requirements based on the actual performance of the produced software artifact.

5) CONCLUSIONS-

As aforementioned, current SE industry still faces the major problems despite the developments in the methods, models, tools and techniques of SE knowledge domain. Although the SE research efforts mostly adopt the principles of quantitative, qualitative paradigms, or both, there are models, such as DSR, which can address the issues in SE research domain. According to our belief and also to the literature review on the theory use in SE research community, there is a need for a process model considering the prescriptions of well-known research paradigms as well as the theoretical and trans-disciplinary foundations of SE as an applied discipline [3]. Consequently, we proposed an extended DSR model, which combined its fundamentals with other approaches, and elaborated on the role of a theory and its constructs. We hope that this model may be a different view to support SE

research community while providing the SE industry with a practice-oriented research and development framework. However, an important limitation of this study is the empirical evidences it needs, and the theoretical support that must be provided by the industry and academicians. As a result, our effort may be considered as a preliminary attempt rather than a complete solution.

REFERENCES:

[1] Y. Wang, "software engineering foundations: A software science perspective", 1st Ed. New York, USA: Auerbach Publications, Taylor & Francis Group, 2008

[2] J. Dominguez, "the curious case of the Chaos Report (2009)", Project Smart, 2009.

[3] M.P. Uysal, "in search of software engineering foundations: A theoretical and trans-disciplinary perspective", Proceedings of the 7th International Conference on Computer Engineering and Technology, Paris, France, April 13-14, 2015.

[4] S. Gregor and A.R. Hevner, "positioning and presenting design science research for maximum impact", MIS Quarterly, 37(2), pp.337-355, 2013.

[5] V.K. Vaishnavi and W.J. Kuechler, "design science research methods and patterns: Innovating information and communication technology". Auerbach Publications, Taylor & Francis Group, NW, USA, 2008.

BIOGRAPHY:

Miss. Pooja Gajanan Wele has completed B.E Degree in Computer Sci. & Engineering from Sant Gadge Baba Amravati University, Amravati, Maharashtra. Pursuing Master's Degree in Computer Sci. & Engineering from P.G. Department of CSE, S.G.B.A.U Amravati

(e-mail id- welepooja@gmail.com)