Abstract:

Presently software is integral to modern society and leads a way in business processes. Software has to follow paths like commencement, process, initial development, productive operation and changeover from one to another generation. A paradigm in engineering process describes flow of activities and disposition of successive phases one after another. A systematic approach like this provides a platform for Analysis, designing, scheduling, testing, configuration management activities, and other software maintenance activities. Under the software development process it is emphasized to undertake a systematic structured approach.

Keywords: Software process, SDLC, waterfall model, prototype model, RAD

1. Introduction

Software engineering concepts are vital and significant for the success of any software. Since early 1960’s, many views have appeared on the different phases that should be the part of a software life cycle[1,2]. Royce in 1970 propagated the basics of waterfall software model. It described the complexity of iterative nature of the software and redoing the phases. Typically a software process is made up of one more of the followings:

Feasible study: This part of software engineering is meant to better understand the customer needs and to further know about the business objectives that motivated a customer to automate or update an existing software system[3,4].

Requirement Analysis and Specification: This specification is a detailed document specifying detailed functionalities[5]. It is about the general problem specification. In fact this document is the base on which any software solution is provided. This describes operational objectives, performance bench marks and resources needed to support maintenance[6,7].

Functional Specification or Prototyping: It is aimed at detailing computational objects desired by a system. To know the objects further, the attributes and relationship among objects are identified. A prototype that manages the objects and functional constraints is explained[8].

Partition and Selection: Structural decomposition of the system into manageable modules is an important activity. It supports reusability and thus minimizes cost etc.

Architectural Design and Configuration Specification: Various architectural designs define the interconnection and interfaces between resources for the modules of the system[10]. It facilitates software design and overall software configuration management activities.

Detailed Component Design Specification:
The functions that transform components from input to desired output are the prerogative of component design.

Component Implementation and Debugging: Developing a source code for implementation of design specification and subsequent debugging is the next activity performed.
**Software Integration and Testing:** Software testing is essential to gain confidence that software performs according to software requirement specification and is error free[11]. Software integration is tasked with ensuring that individual modules deliver the intended function when they are put together.

**Documentation Revision and System Delivery:**
All functions and procedures need to document and prepare a user guide in helping the system better.

**Deployment and Installation:** Once the software development activity is over it needs to be deployed at customer’s end. Deployment and Installation is performed to ensure bug free systematic operation of software[12].

**Training and Use:** A team of trained officials carry out the guidance and provide useful training at customer’s end.

**Software Maintenance:** Once the system is in operation, the biggest challenge is managing the constant changes which occur from time to time. Software maintenance activities are a set of activities which manage this[13].

2. **What is a software life cycle model?**

The life cycle of software illustrates the detailed understanding about the software and the methodology suitable for the software development. There are two school of thought a. Descriptive and b. Perspective. The former illustrates the manner in which the software was developed and the later emphasizes the manner in which software should be developed[14]. Illustrative approach is beneficial for improvement in software development methods followed whereas perspective approach is for broad framework about development process.

2.1 **What is a software process model?**

The process of software development focuses on the detailed activities needed to be carried out to meet the software requirements. Such activities are useful in developing the above mentioned tasks[15].

3. **A new approach to software development: The framework**

3.1 **Relevance**

Software engineering tasks are focused at problem identification and the challenges involved in providing solutions. The objective is to increase knowledge about problems at hand and its possible solutions. “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 domains, all stakeholders of a software project have to justify the problem, and also define the solution and acceptance criteria for that problem”.

3.2 **The Type of Research**

The research on Software engineering has many different methods and the main focus lies on analysis, design, implementation and maintenance. One important aspects type of research is theory based or practical experience based.

A Software Engineering 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.

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. 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.

3.3 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.

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. 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.

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. 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. It is indicated that this 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 principles are based on framework for evaluating the knowledge and research contribution of a research study in SE discipline. 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 non-trivial extension of a known solution to a new problem results in “expatiation” 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. The skillful selection and application of appropriate theories is also important for constructing and evaluating the software artifacts. In addition to the results of new technique 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.

3.4 Design-Build-Test Cycle

The main activities of this phase are
• Elaborating on theory constructs
• 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. 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.

3.5 Evaluation & Conclusion:

At this phase, the evaluation of the software artifact is done with single or multiple case studies 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.

References

2. Clifton C., Kantarcioglu M., Vaidya J., Defining Privacy for Data Mining. Purdue University, West Lafayette.
5. Hann J., Kammer M., Data Mining concepts and techniques 2ed.
11. Pinkas B., Cryptographic Techniques for Privacy-Preserving Data Mining SIGKDD Explorations, the newsletter of the ACM Special Interest Group on Knowledge Discovery and Data Mining, January 2003,
12. Sweeney L., Achieving k-anonymity privacy protection using generalization and suppression. 2002 CMU.
