Scholars Journal of Engineering and Technology (SJET) **ISSN 2321-435X (Online)**
Sch. J. Eng. Tech., 2016; 4(1):40-48 **ISSN 2347-9523 (Print)**
©Scholars Academic and Scientific Publisher
(An International Publisher for Academic and Scientific Resources)
www.saspublisher.com

Original Research Article

A Comprehensive Contextual Industrial Engineering Research and Software project Management Using Product Metrics

Prodip Kumar Sarker, Md. Jamal Hossain
Department of Computer Science and Engineering, Begum Rokeya University, Rangpur, Bangladesh.

***Corresponding author**
Prodip Kumar Sarker
Email: sarker.bsmrstu@gmail.com

Abstract: For the purpose of exact and flawless valid conclusions while aggregating evidence of software project, it is important to describe the context in which industrial studies and inspection were conducted. This paper exhibits and structures the context for comprehensive and empirical inspections, studies, analysis and provides a checklist. The prime concern of this paper is to assists decisions makers and researchers in making well equipped informed decisions concerning which parts of the context is to include into the project descriptions and software project management and maintenance using product metrics. In additions to this there will be surveyed the descriptions of industrial studies and analysis.
Keywords: Contextual Structure, Project Checklist, Software Project, Product Metrics, Project Management, Project Development.

INTRODUCTION

To assists the decisions makers and researchers in defining the context in a way to allow better aggregation of evidence, this paper proposes the following noble contributions: (1) provides a checklist to describe the context in industrial studies and analysis; (2) conduct investigations, inspection and a brief survey to determine the degree of industrial case studies. (3) Providing the software selection metrics features.

Overview of the Approach

Software Engineering is concerned with the acquisition, definition, management, monitoring, and controlling of software development projects as well as the management of risks emerging during project execution. The research for Software Design & Architecture advances techniques for the development, management, and analysis of (formal) descriptions of abstract representations of the software system as well as required tools and notations.

In order to successfully develop and maintain a large software project for a long times even for a many years, organizations need to keep some specific procedures and modules which will not become much more complicated [4]. They need to keep up improving the maintainability of each module by cleaning up spaghetti codes, reconfiguring module structures, redesigning functional logic and data handling, etc.

After all, the maintenance activities greatly rely on individual engineer's knowledge and experience about how the software has been changed. Especially in Japan, many software companies used to have a person called *walking dictionary* – an expert engineer who have been maintaining legacy software for more than a decade and knows various features concerning to that software, e.g., current specification of the software, mapping between specification and actual source code, how complicated each software module is, etc. Such a maintainer can properly detect the software module that should be reengineered, and can keep every module not to become too much complicated through continuous reengineering activities (sometimes such activities are called *refactoring*) [1,5]. Till recent years, walking dictionaries had stayed in one company until they become the retirement age, and they took care of the company's software; and thus, the maintenance cost was kept low for many years.

The approach of this project is to analyze a large software system that has been successfully developed and maintained. We always keep in mind that the succeeded project is appropriately developed and maintained, i.e., procedures in the development and decisions have been made properly.

Context Documentation Regarding the Checklist

The checklist is structured in six different context features, namely product, processes, practices and techniques, people, organization, and market (see

Figure 1). In the center of the context features the object of the study is shown. The object of the study interacts with the context. For example, when having an agile process as the object of study, the process is used to develop a product, is executed by people, interacts with other processes, and is supported by practices, tools and techniques. Furthermore, the object of study is embedded in an organization. The organization is operating within a market [6].

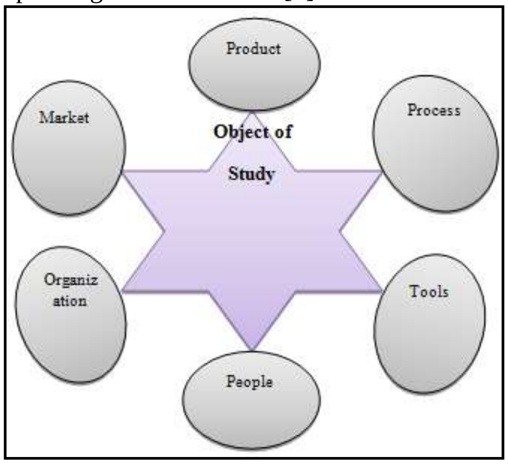

Fig-1: Context features

Each context features comprises a set of context elements describing the context. In the following this paper provides a description of the features and examples of related elements.

Product:

The product is the software system developed with the help of the object of study. Context elements:

- **Maturity**

The maturity of the product needs to be described. For example, by saying how long it was on the market, how many releases there were, etc.

- Quality:

The product development is driven by different quality aspects (e.g. a development effort is undertaken to increase maintainability of the product).

- Size:

Size of the product measured in, for example, lines of code, or number of function points. The size is an important indicator for product complexity.

- System type:

The system can be of different types, such as an information system, embedded system, web application, or distributed system.

- Customization:

This means that the product is either general or customizable and can be tailored to different market segments.

- Programming language:

This element describes the programming language in which the system was developed.

PROCESSES

The process describes the work-flow of the development. Context elements:

- Activities:

The activities are different steps in the development process (e.g. specifying requirements).

- Work-flow:

The work-flow describes the order in which activities are executed (including branching, merging, iterations, etc.).

- Artifacts:

Artifacts are the results of the activities (e.g., the requirements specification).

Practices, Tools, Techniques:

Practices, tools, and techniques describe systematic approaches that are used in the organization, and are interacting with the object of study. Context elements:

- CASE Tools

This element describes tools that are used to support or automate software development (for example, integrated development environments, automated test tools, etc.).

- Practices and Techniques:

This can be systematic approaches interacting with the object of study. For example, when studying an agile process practices could be time-boxing, frequent integration, or pair programming.

People

The human factor is very important when studying software development, as it has a major impact on the success of software development. Thus, the factor has to be covered in the context. Context elements:

- Roles:

This element describes what type of roles is involved in using the object of study. This includes a description of the main responsibilities and authorities associated with the roles.

- Experience:

Experience is concerned with the areas that people affected by the object of study have worked in, and for how long. Furthermore, education and trainings are of interest.

Organization

The organization describes the company structure in which the other context facts and the solution are embedded in. Examples for elements are:

- Model of overall organization:

The organization model describes how the company is organized, such as matrix-organization or hierarchical organization. Furthermore, it could be discussed whether the organization is flexible or strict.

- organizational unit:

The organizational unit is a part of the organization closely interacting with the object of study. For example, this can be a project (temporary existing unit) or department (permanent unit). The organizational unit can be complemented with management related information such as responsibilities of the unit, and size measured as number of persons involved.

- Certification:

This tells whether the organization is certified.

- Distribution:

The organizational units are either collocated or distributed (nationally or internationally).

Market

The market represents the customers and competitors. Elements describing the market are:

- Number of customers:

A contract is established already between developer and customer. Market-driven development targets a large and open market of potential customers that might buy the product after release.

- Market segments:

The market segments describe groups of customers that share a common need.

- Strategy:

The strategy describes how to address the market in the long-term. For example, the company can run a niche-strategy where they compete with a special product (e.g. in terms of extremely high quality) or a strategy to compete on a low price.

- Constraints:

The market can put constraints on software development, such as a very short time-to-market, or certifications.

Software product Selection metrics

Software selection is an important and most crucial activity in the development of software project. The project team must set up yardstick for the selection. In this phase this paper has described the following feature for the selection.

Reliability

This feature ensures that the software will execute for a specified time period without failure. It also relates to the ease of recovery and ability to give consistent and optimum results.

Functionality

Functionality defines the facilities, performance and others factors that the users require in the final product.

Capacity

Capacity planning is the process of determining the production capacity needed by an organization to meet changing demands for its products.

Flexibility

It is a measure of the effort required to modify the operational program. One feature of flexibility is adaptability, which is a measure of the ease of extending the product.

Usability

Usability refers to that characteristics the product is user friendly and ease of access.

Security

Software security assurance is a process that helps design and implements software that protects the data and resources contained in and controlled by that software. Software is itself a resource and thus must be afforded appropriate security.

Performance

Deliver robust, consistent service quality to customers and business users, need effective ways to monitor and optimize the performance of business-critical workloads. Track performance and utilization trends, discover potential issues before they occur, minimize the mean time to recovery (MTTR), boost the efficiency of our environment and deliver the responsive experience users expect.

Serviceability

This feature focuses on the documentation. The complete documentation is critical for software enhancement.

1. **Table-1: Software product Selection metrics- A Summary**

2. **Feature**	3. **What to mean**
4. Reliability	5. Delivers optimum and consistent results.
6. Functionality	7. Functions to standards.
8. Capacity	9. Satisfied volume requirements.
10. Flexibility	11. Adapts to variable needs.
12. Usability	13. Easy to operate and understand.
14. Security	15. Prevent unauthorized access and maintains integrity.
16. Performance	17. Properly delivers as expected.
18. Serviceability	19. Well documentation.
20. Ownership	21. Right to modify.

Data for Analysis:

The traditional functional decomposition and data analysis design approach measure the design structure and/or data structure independently, object-oriented metrics must be able to focus on the combination of function and data as an integrated object [1]. The evaluation of the utility of a metric as a quantitative measure of software quality was based on the measurement of a software quality attribute. The metrics selected, however, are useful in a wide range of models. The object-oriented metric criteria, therefore, are to be used to evaluate the following attributes:

- Efficiency - Are the constructs efficiently designed?
- Complexity - Could the constructs be used more effectively to decrease the architectural complexity?
- Understandability - Does the design increase the psychological complexity?
- Clarity-Does the instruction is clear and lucid?
- Testability - Does the structure support ease of testing and changes?
- Maintainability-Does the structure can easily be maintained?

Software development

Software development is the process of computer programming, documenting, testing, and bug fixing involved in creating and maintaining applications and frameworks involved in a software release life cycle and resulting in a software product. It also refers to a process of writing and maintaining the source code, but in a broader sense of the term it includes all that is involved between the conception of the desired software through to the final manifestation of the software, ideally in a planned and structured process [1]. Therefore, software development may include research, new development, prototyping, modification, reuse, re-engineering, maintenance, or any other activities that result in software products [2].

Software can be developed for a variety of purposes, the three most common being to meet specific needs of a specific client/business (the case with custom software), to meet a perceived need of some set of potential users (the case with commercial and open source software), or for personal use (e.g. a scientist may write software to automate a mundane task). Embedded software development, that is, the development of embedded software such as used for controlling consumer products, requires the development process to be integrated with the development of the controlled physical product. System software underlies applications and the programming process itself, and is often developed separately.

The need for better quality control of the software development process has given rise to the discipline of software engineering, which aims to apply the systematic approach exemplified in the engineering paradigm to the process of software development.

A software development process (also known as a software development methodology, model, or life cycle) is a framework that is used to structure, plan, and control the process of developing information systems. A wide variety of such frameworks have evolved over the years, each with its own recognized strengths and weaknesses. There are several different approaches to software development: some take a more structured, engineering-based approach to developing business solutions, whereas others may take a more incremental approach, where software evolves as it is developed piece-by-piece. In this paper most methodologies that shares some combination of the following stages of software development:

- Analyzing the problem
- Market research
- Gathering requirements for the proposed business solution
- Devising a plan or design for the software-based solution
- Implementation (coding) of the software
- Testing the software
- Deployment
- Maintenance and bug fixing

These stages are often referred to collectively as the software development lifecycle, or SDLC. Different approaches to software development may carry out these stages in different orders, or devote more or less time to different stages. The level of detail of the documentation produced at each stage of software development may also vary.

Software Development Activities

To develop software the following activities are necessary.

Identification need, Planning, Designing, Implementation, Testing, and Documentation.

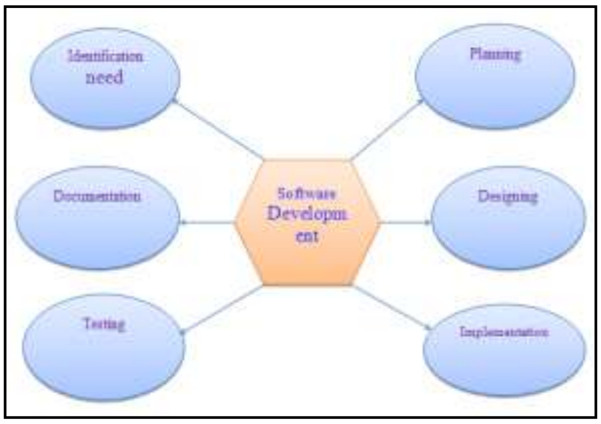

Fig-2: Software Development Activities

Context in Industrial Studies and Analysis

In the case of software industry, the context selection is a very crucial and important key factor for the improvement and for the successful project development.

Context is time dependent i.e., it can be changed in course of time. So in the beginning of the project development it should be carefully select the accurate and appropriate context that can suit for the successful project development in the most part. In the software industry, a special team is involved in detecting the suitable context.

In this work, we have tried to give an effective and an efficient methodology called CMS (Combined Method Strategy) that address some skills in the software system analysis and studies. The following are the most important that are suggested to follow-

Teamwork

The process of working collaboratively with a group of people in order to achieve a goal. Teamwork is often a crucial part of a business, as it is often necessary for colleagues to work well together, trying their best in any circumstance. Teamwork means that people will try to cooperate, using their individual skills providing constructive feedback, despite any personal conflict between individuals.

Communication

Communication is simply the act of transferring information from one place to another. Although this is a simple definition, when we think about how we may communicate the subject becomes a lot more complex. There are various categories of communication and more than one may occur at any time.

Mentoring

Mentoring is a process for the informal transmission of knowledge, social capital, and the psychosocial support perceived by the recipient as relevant to work, career, or professional development; mentoring entails informal communication, usually face-to-face and during a sustained period of time, between a person who is perceived to have greater relevant knowledge, wisdom, or experience (the mentor) and a person who is perceived to have less.

Decision making

Decision making is the process of making choices by setting goals, gathering information, and assessing alternative occupations.

Negotiating

Negotiation is a method by which people settle differences. It is a process by which compromise or agreement is reached while avoiding argument and dispute.

Information gathering

Information gathering refers to gathering information about the issue we're facing and the ways other organizations and communities have addressed it.

Resource managing

Resource management is the efficient and effective development of an organization's resources when they are needed. Such resources may include financial resources, inventory, human skills, production resources, or information technology (IT).

Defect detecting

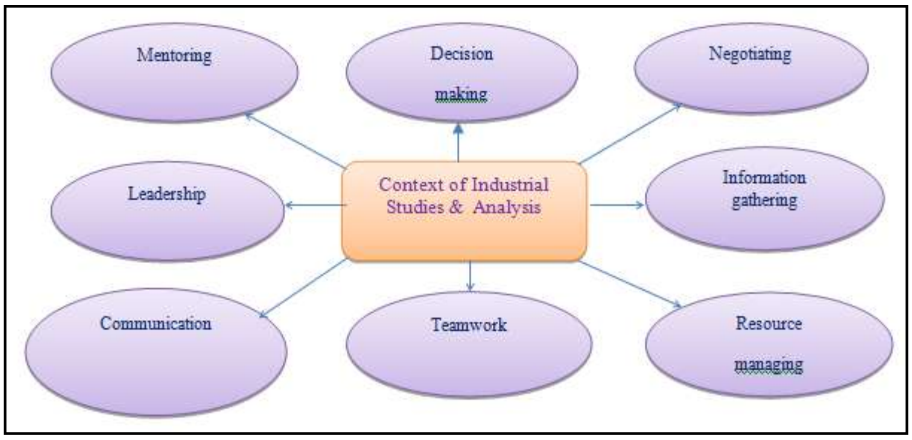

Fig-3: Block diagram of Industrial Studies and Analysis.

Contextual design

Design describes a final system and the process by which it is developed. In others words it can be defined as a process of a developing the technical and operational specification of a candidate system for implementation. It is regarded as a blueprint for the system development.

Definition of industrial design

Industrial design studies function and form—and the connection between product, user, and environment. Generally, industrial design professionals work in small scale design, rather than overall design of complex systems such as buildings or ships. Industrial designers don't usually design motors, electrical circuits, or gearing that make machines move, but they may affect technical aspects through usability design and form relationships.

A checklist for Context Documentation

A software project metric is a standard of measure of a degree to which a software system or process possesses some salient property. Even if a metric is not a measurement (metrics are functions, while measurements are the numbers obtained by the application of metrics), often the two terms are used as synonymous. Since quantitative measurements are essential in all sciences, there is a continuous effort by computer science practitioners and theoreticians to bring similar approaches to software development. The goal is obtaining objective, reproducible and quantifiable measurements, which may have numerous valuable applications in schedule and budget planning, cost estimation, quality assurance testing, software debugging, software performance optimization, and optimal personnel task assignments.

A metric can be *directly collected* through observation, such as number of days late, or number of software defects found; or the metric can be *derived* from directly observable quantities, such as defects per thousand lines of code, or a cost performance index (CPI) [8]. When used in a monitoring system to assess project or program health, a metric is called an indicator, or a key performance indicator (KPI).

In this paper a theoretical method named MMM (Metric Management Means) which helps the people specially the researchers in the most par to identify and track the strategic objectives. Different types of projects will require different types of metrics—a software development project will call for different measurements than, say, a merger and acquisition transition project.

The following table defines the criteria of the most common feasible strategic measures people want to be updated about:

Table-2: The criteria for strategic measurement

Strategic measure	Questions to be answered
Time	How are we doing against the schedule?
Cost	How are we doing against the budget?
Resource	Are we within anticipated limits of staff-hours spent?
Scope	Have the scope changes been more than expected?
Quality	Are the quality problems being fixed?
Action Items	Are we keeping up with our action item list?

A common saying about metrics is: “If it cannot be measured, it cannot be managed.” Clearly the lack of metrics can make it harder for a project manager to do the best job possible. At the same time, metrics are useful only if they are just that – useful.

Software maintenance

Software maintenance in software engineering is the modification and enhancement of a software product after delivery to correct faults, to improve performance or other attributes.

A common intuition of maintenance is that it rather involves fixing up defects and deficiencies. However, one study indicated that over 80% of maintenance effort is used for non-corrective actions [3].

The key software maintenance issues are both managerial and technical. Key management issues are: alignment with customer priorities, staffing, which organization does maintenance, estimating costs. Key technical issues are: limited understanding, impact analysis, testing, and maintainability measurement.

Project Maintenance Tactics

The project maintenance task is so easy in reality. Different software organizations follow different strategy to maintain the project activity. Project maintenance is an elegant task that ensures the longevity of any system. The concepts and ideas can change in course time but the maintenance techniques can hold that product many years. So any work done to change the software after it is in operation is considered to be maintenance work. The purpose is to preserve the value of software over the time. The value can be enhanced by expanding the customer base, meeting additional requirements, becoming easier to use, more efficient and employing newer technology. Maintenance may span for 20 years, whereas development may be 1-2 years.

In this paper some of the effective maintenance tactics that is directly benefitted to the software project management system. As Software maintenance is a very broad activity, the following tactics can be considered in the maintenance phase that includes-

- Error correction,
- Review,
- Enhancements of capabilities,
- Deletion of obsolete capabilities,
- Optimization,
- Evaluation,
- Controlling,
- Making modifications.

Inspection and Investigation

An inspection is one of the most common sorts of review practices found in software projects. The goal of the inspection is for all of the inspectors to reach consensus on a work product and approve it for use in the project [7]. Commonly inspected work products include software requirements specifications and test plans. In an inspection, a work product is selected for review and a team is gathered for an inspection meeting to review the work product. A moderator is chosen to moderate the meeting. Each inspector prepares for the meeting by reading the work product and noting each defect. The goal of the inspection is to identify defects. In an inspection, a defect is any part of the work product that will keep an inspector from approving it. For example, if the team is inspecting a software requirements specification, each defect will be text in the document which an inspector disagrees with.

The Inspection process

The process should have entry criteria that determine if the inspection process is ready to begin. This prevents unfinished work products from entering the inspection process. The entry criteria might be a checklist including items such as "The document has been spell-checked".

The stages in the inspections process are: Planning, Overview meeting, Preparation, Inspection meeting, Rework and Follow-up. The Preparation, Inspection meeting and Rework stages might be-

- **Planning:** Planning is the most important step in the system inspection. The inspection is planned by the moderator. Actually, planning propels the inspection commencement.
- **Overview meeting:** The key area under inspection mainly describes the background of the work project. The author describes the background of the work product.
- **Preparation:** Each inspector actually who conducts the inspection examines the work product to identify possible defects.
- **Inspection meeting:** During this meeting the reader reads through the work product, part by part and the inspectors point out the defects for every part.
- **Rework:** The author makes changes to the work product according to the action plans from the inspection meeting.
- **Follow-up:** The changes by the author are checked to make sure everything is correct.

The process is ended by the moderator when it satisfies some predefined exit criteria.

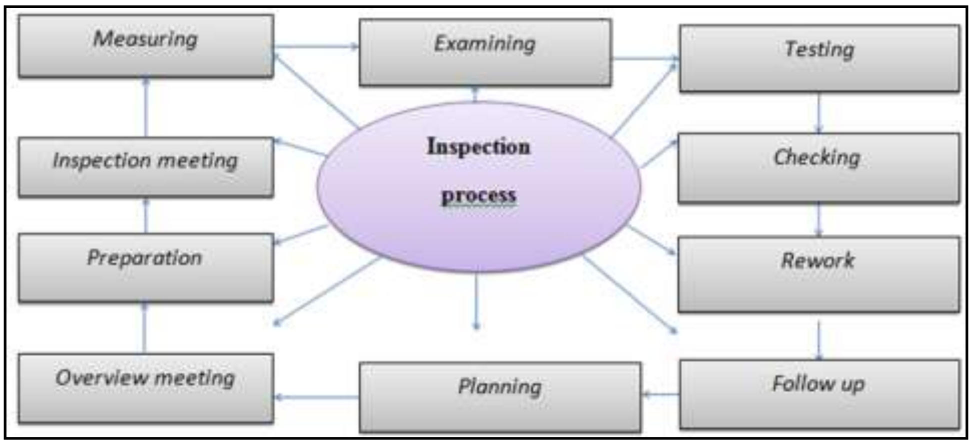

Fig-3: The inspection process cycle

Inspection roles

Inspection roles have an important aspect in the software industry. Inspection is used for the purpose of determining if a body is complying with regulations [7]. This process examines the criteria and talks with involved individuals. A report and evaluation follows such experiments**.**

During an inspection the following roles can be used.

- **Author:** The person who created the work product being inspected.
- **Moderator:** This is the leader of the inspection. The moderator plans the inspection and coordinates it.
- **Reader:** The person reading through the documents, one item at a time. The other inspectors then point out defects.
- **Recorder/Scribe:** The person that documents the defects that are found during the inspection.
- **Inspector:** The person that examines the work product to identify possible defects.

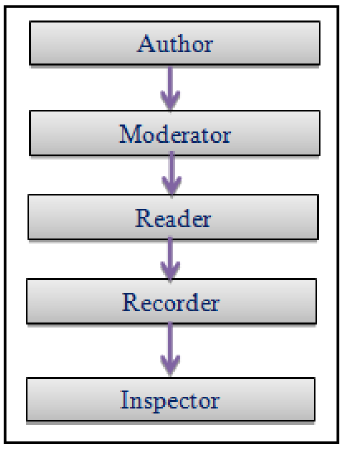

Fig-4: Inspection roles flow diagram

Industrial Studies Context

Table-3: Report on Research aspect considering object of Investigation.

Aspect	**SR(8)**	**SP(5)**	**QA(5)**	**SC(4)**	**UML(2)**	**RM(3)**	**RES(2)**	**AE(1)**	**CS(3)**	**Total(33)**
Product	6	3	5	2	1	1	1	1	1	23
Process	3	2	3	1	1	0	0	1	2	13
Tool	4	2	4	3	0	3	2	1	1	20
People	0	1	1	0	2	0	0	1	0	5
Organization	0	5	3	4	2	1	0	1	3	19

Conclusion

This paper mainly highlights the product matrices for the description of context design and system development. In this paper there is also described an efficient and effective methodology that focuses the basic skills (Teamwork, Communication, Leadership, Mentoring, Decision making, Negotiating, Information gathering, Resource managing, Defect detecting) for software analysis and studies. There is also presented a theoretical review of industrial studies that studies investigating a similar object do not agree on which context facets are important to mention [7]. The checklist aims to help researchers to take informed decisions on what to include and not to include. The checklist also serves as a basis for discussion to reach a consensus in different communities which context elements are most important for their focus area. In future work the checklist has to be further extended. The viewpoint also plans to do in-depth analysis of a selection of articles focusing on context facets as well as elements.

REFERENCES

1. Kitchenham B, Pfleeger SL, Pickard LM, Jones PW, Hoaglin DC, El Emam K, Rosenberg J; Preliminary guidelines for empirical research in software engineering. Software Engineering, IEEE Transactions on, 2002; 28(8): 721-734.
2. Runeson P, Höst M; Guidelines for conducting and reporting case study research in software engineering. Empirical software engineering, 2009; 14(2): 131-164.
3. KitchenhamBA, Dyba T, Jorgensen M; Evidence-based software engineering; in proceedings of the software engineering (ICSE 2010), 2010; 273-281.
4. Glass RL, Vessey I, Ramesh V; Research in software engineering: An analysis of the literature, Information and Software technology; 2002; 44(8): 491-506.
5. Barry B, Clark B, Horowitz H, Westland C, Madachy R, Selby R; Cost models for future software life cycle processes: COCOMO 2.0, Annals of Software Engineering; 1995; 1(1):5794.
6. Runeson H, Höst M; Guidelines for conducting and reporting case study research in software engineering; Empirical Software Engineering, 2009; 14(2): 131-164.
7. Petersen K, Wohlin C; Context in industrial software engineering research; Proc. the 2009 3rd International Symposium on Empirical Software Engineering and Measurement, IEEE Computer Society, 2009; 401-404.
8. Paul C, Connor RVO'; The situational factors that affect the software development process: Towards a comprehensive reference framework; Information and Software Technology, 2012; 54(5): 433-447.