IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 17, Issue 5, Ver. IV (Sep. – Oct. 2015), PP 88-95
www.iosrjournals.org

A Review on Systematic Quality Software Designing and DevelopmentPractices

Mulugu Narendhar[1], Dr.K.Anuradha[2]
[1]*Assoc.Prof & HOD, CSE Dept, Scient Institute of Technology, Hyderabad,TS,India*
[2]*Professor & HOD, CSE Dept, GREIT, Hyderabad, TS, India*

Abstract: *Software Engineering process, technology and the results cover the definition of suitable models for the environment to ensure its quality.Any software development project is very important to create quality software design artifacts wheresoftware architecture and design rules set plays an important part.The current program and designing practice has been provided by the use of proper testing and code created significant improvement in quality and productivity.This paper presents a review on the systematically and efficiently designing, modelling and verification processto help the improvisation of the quality of software designing process and development practices through various characters, roles and measures to participate in testing.*

I. Introduction

Quality of software designing and development a successful process management planning, measurement, and control is required. Program development generally needs the operating range of the programming process to define the translation functions each has its own exit criteria. A measurement should be there for the perfection of product in its development at any time for the next inspection or test. Finally, the process of measuring data should be used for the controlling process.This approach not only conceptually interesting, but the large and small projects, systems and applications programming holds several programs have been successfully applied and enabled testing for the improvisation of the quality of products.The purpose of this paper is to provide an insight on the planning practices, measurement and control functions for software quality designing and development.

The complexity of software designing, managing and development gives rises to many problems to meets the quality,budget and timeline as it an expensive business. Here both manpower and technology are necessary resources and also require a lot of investment. Many projects have been completed with errors and customer dissatisfaction which are installed due to the low-quality products delivery. These are the main issues that have been trying to solve for many years in all the software industry.Today a significant impact on the software business on the economy [10] and more cost effective production generates considerable interest to software development and software quality. The current software industry is suffering due to trouble in producing good quality software [4] and ineffective development policies [21] which cause towards the inadequate losses.

To manage the inherent complexity of software quality development we needs a various systematic software designing method and disciplinary software development to solve the problem with its strengths and weaknesses for the development and ways to installation in each application domain. This paper tries to present a review on this issues related to software engineering, designing, modelling and management practices.This following paper organized majorly in five sections. Section-2 presents an insight on the software engineering process, Section-3 discuss about the software quality, metrics and techniques process. Section-4 presents software design and development modelling approaches and Section-5 presents the related works related to software quality of service and development improvisations and finally Section-6 presents the conclusion.

II. Software Engineering Process

Software engineering in computer science refers to the methodology and procedural control in software development. The more complexity raisein softwaredevelopment to occur due to the developer team sizes grows, many software engineering process models, ensuring quality,followsdeadlines and also to meet the end-user expectation. This all process need to performin the software engineering process by the team to becoming more applicable and relevant.Special stages and tasks recommended that the process to be completed in the model development process.Checklist areprovidedat each stage of the development process need to be completed before moving on to the next successful step. By following the modelling process for the development team can greatly provide the high-quality software whichdemonstratesthe compliance with regulatory standards to meet the deadlines and team capability.There are several software engineering process models which includes the Waterfall, Spiral and Agile procedures.They have evolved to meet the various requirements and generally accepted meaning in a different case and the associated benefits. However, software engineering should be aware of all the forms and comply with all the fundamental development processes and functions are important parts of the actions. The configuration management, requirements gathering, architecture, development, testing and deployment are being included in this process.

III. Software Quality Process

Software Quality engineering process (SQEP) evaluates the assesse and improved software quality in software development. It is often as a result of software quality software engineering satisfied that action, performance and interface requirements with a change of software in contrasts to the reliability, maintainability, transportability defined to meet the requirements.It must be built into software products to complete the development in quality for established quality requirements.

SQEP ensure the software quality process which includes taking the proper place, and the resulting software product that meets the quality requirements. The investigation showed the degree of functional requirements of the quality compliance requirements in general, should be determined by analysis. Software Development engineering performs complementary function to demonstrate the functional requirements.Their common objective to ensure safe, reliable and the development of quality software engineering products.

A. Software Quality

Software Quality engineering processare in needs to be chosen the requirement sets for the qualities to be evaluated. Some commonly used qualities for quality evaluationsare reliability, maintainability, interoperability, testability, usability, traceability, reusability, sustainable and efficiency.Some of the mainqualities process are discussed below.

- **Reliability**

Software reliability is define as the performance of a programperform in given period of time to achieve the required accuracy for an amount of work that can be done. It is mainly concerned about the correct and identify software defects.Moreover, it is unknown software bugs which are implemented in hardware and data environment whichcompensate to techniques and concerned to the software execution.A similar idea is useful for software, although failure mechanisms are different and the mathematical predictions for the hardware used have not yet been applied to useful software.

- **Maintainability**

Software maintainability is defined as a process to discover the software errors and provide the correcting convenience approach or procedures in the software. Directly measuring the maintainability of the software remains no efficient way to predict, but there is a significant level of knowledge about the software features an easy software maintenance. The modules, self (internal) documentation, code readability, and a structured coding techniques are included. These same qualities also, to improve the ability to increase the sustainability and improvisation of the software.

- **Interoperability**

The ability to exchange information and share mutual interactive Software, two or more software systems have the ability to use the information.

- **Efficiency**

Software efficiency is the amount of resources needed to fulfil its minimum hardware functions.Many software qualities are there for the efficiency improvement, but some of them have no work done to improve the assessment, or morespecific system software will be important.Some properties may lead to fewer improve if others are maximized. For example, a piece of software, it may be necessary to write parts in assembly more efficient, but this will reduce software transportability and maintainability.

B. Quality Metrics

Quality Metrics on the question of the quality or the quality of the design or code of the software is usually calculated from the quantitative measurement of some attribute values are. Several measures have been invented, and many have been used successfully in certain areas, but none has been approved widely.

C. Software Quality Engineering Process

There are two software features, reliability and maintainability command the most attention. Measurable or predictable, although they are few experimental programs and methods, software was developed to improve the reliability and maintainability. SQEP may be included in the program for a variety of activities in terms of these two features are described. Additional features of this activity could be used as a model for the SQEP activities.

- **Qualities and Attributes**

In setting out a SQEP program that is being developed at an early stage to select the features that are important in the context of the use of software. For example, the flight software, reliability and efficiency are usually the highest priority features. If the link-up can be revised at the time of the flight software, flight, feed may be of interest, but as a place to run or not to drive design considerations. On the other hand, the science analysis software and the ability to change and maintenance should be ease of use with reliability concern.

Software features selected and ranked, then noted that certain features of the software that will help to improve the properties. For example, there is a feature of the ways to increase both reliability and maintainability. Modular software for small, self-contained, functionally designed to result in code that is reduced into single components or units. Because of the code easier to understand the interaction between the units easy to maintain modular code, and the low-level functions are a few units of code. It is entirely a small, self-contained unit, modular code is easier to test, even more reliable.

So all the software features of the measurable attributes of design and code, and the quality can be measured easily, so it will be easy. The idea is to select the features or want to measure, analyze, design and code, or observable properties is launched. Information hiding, the strength properties, such as binding, and should consider in combining.

- **Quality Evaluations**

Once decisions have been made about some of the features and software quality objectives, quality can be estimated. The objective of the research is to promote the software to produce the desired properties is a standard or measure the effectiveness of the process. For example, one should undergo assessment of the quality of the design and coding standards. If you want modularity, the standards are clear and should set the standard for the units or the size of the components. So the link to maintain internal documentation, documentation standards required internal documentation should be clear and good.

Designs, Models and codeshouldbe evaluated for the quality needs. This can be a part of the walkthrough or the inspection process, or can be of a quality audit. In either case, the standard running and good software engineering practices against the examiner's knowledge of the cost, and the production of poor-quality examples identified for correction.

- **Fault Tolerance Engineering**

The software should have high reliability, fault tolerance, indicating a need to set up. It should offers and determine which critical functions and software requirements must be met. The software, engineering activities and the need to determine the reliability or any of the methods should be developed to ensure fault tolerance can be achieved. There are some of the techniques that have been developed for high reliability environments: the input data checking, error tolerance, proof of correctness and autonomous development [21].

D. Techniques and Tools

There are also useful tools for measuring and evaluating quality software engineering process. They are the system and software simulators which are based on the pattern of behaviour of the system. Most software tools are used to compute the metrics used in the form of code, or the code to recognize the dynamic analysers. For example, limitation of the portability of the system to detect a host of devices to help decides which calls for a special purpose.

IV. Software Design And Development Modelling

Software design is adiscipline ofsoftware development cycle [25]. This consists of implementation and continued management of engineering design and description of these requirements is in the early stages of development. It provides software engineering and steadily for a number of activities in the process of specification, implementation, quality assurance, coordination, management and control of the development, and methods for effective self-support software development techniques, combined. It mainly refers to the process of developing such a model which is usually a software development life cycle model.

Software development describes the process of software designing and modelling. Software modelling format (e.g. user stories, use-cases and requirements documents), graphic formats (e.g. class diagrams, sequence diagrams), and other forms (e.g. mathematical models) are included [22]. Often overlooked is that the software model, and right in front of everyone's eyes, is the root code. Source for the application code is another example, it can be implemented just one (i.e. directly or can be interpreted) and the software development primary objective.

A. Life Cycles

Classic stages of the software development life cycle model is to be describe the steps or stages, even if it is introduced in the field of software engineering called as "waterfall" model [X]. This design requirements analysis, specification, design, implementation, testing, operation and maintenance phases of the mark, and they are well-defined deliverables at the end of each stage assumes that occurs in a strict sequential order. Milestones in the development of the projects they are managing the deliverables specification documents and the implementation of the code.

Many alternatives models are exist in compete to the waterfall model. Rapid prototyping is on such alternate based on an element of uncertainty that may be useful in situations where a substitute is required. Model is a model to be built in the system, and may be used by a developer who works with the customer to determine requirements. It can also be used to test the viability of a proposed system, and can sometimes even be regarded as the ultimate system of the partial specification. A sample is usually modified and adapted several times, and therefore less thorough, walk-through of the resurrection of the classic waterfall model is to be regarded as one of the steps.

Any software development process, in general the requirements analysis, design, implementation, test and maintenance parts. The famous waterfall model of software development life cycle is given below in Figure -1.

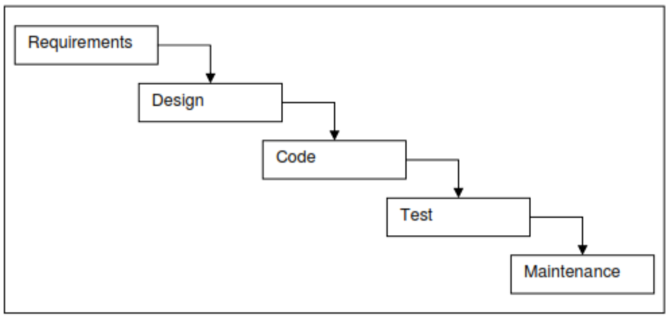

Fig.1: Waterfall Model

The software is divided into two stages such as Software ArchitectureDesign(SAD), and Software Detailed Design (SDD) in the design process. Architecture design is usually referred to as the high-level design.

It functions as software components of these components, their deployment, and operation of deciding on allocation. In SDD, the elements, and interfaces are to meet the needs of their work are more detailed [11].

B. Methods

The complexity of today's software systems to the development of their well-defined, systematic practices to use. Software development processes are engineered to fulfil this role[25].Specifying a software development method and / or the development of a software system refers to a set of policies and guidelines. A method of inquiry to all or part of a Software Development Life Cycle.Thus, there are requirements engineering methods such as Software Requirements Engineering Methodology (SREM) [26], design methods such as Structured Design (SD) [27], and software development methods that include both analysis and design steps such as the Structured Systems Analysis and Design Method (SSADM) [28].Sometimes, however, the term methodology, software engineering literature in terms of the approach and methodology used interchangeably in most of the techniques is used to describe a collection or the system, and there often are confused.

Also, a method in which one or more notations to extend the guidelines developed by the software. These system requirements, design and implementation of the mandate expressed. They also created a software development project is used to describe a wide variety of products and documents. In fact, the concept of a method of expressing ideas about the basic tool, which is inextricably equations, and of the characteristics, the value of a system has been added.

Software engineering methods have been developed over the past 25 years, and this is invariably different notations for specifying and developing software systems and procedures have been deployed. The use of these methods are representative of the diversity of equations prescriptive practices and guidelines.

- **Structured Analysis and Design Technique (SADT)**

Structured Analysis and Design Technique (SADT) [29] using the language in a systematic and structured way of specifying systems, single, rich, highly graphical language combine. Specifications decomposed into smaller units that are easier to grasp the necessity of language.

Structured Analysis (SA) [29] –is the normally SADT subordinate the general philosophy and methods ofany value expressed in six or fewer pieces must be something about giving suggests. The system and its context, a hierarchical, top-down decomposition SADT results in the application of this advice. A collection of interconnected sketches of a model in SA, and multiple models are supported. Each model is a case in point of view and have the benefit of its trend [22].

- **Controlled Requirements Expression (CORE)**

Controlled Requirements Expression (CORE) [30], in order to specify and analyze system requirements, four different drawing views, and seven different stages of the method it uses structured text. The first phase (problem definition) of a customer's authority, business objectives, the problems of the current system, the new system features, future expansion, identifies the costs and time scales. This is explained in terms of matter in the form of a structured.

- **Constructive Design Approach (CDA)**

Structural Design Approach (CDA) [31],in addition to the policy for the development of distributed systems, structured approach rather than the means of notation. From the functional descriptions of the individual components of the method, a set of components and their connections, the system aims to guide the development of the distribution system through the construction of the separation. This argues that its designers, description, structure and evolution of the facilities of distribution systems.

- **Jackson System Design (JSD)**

Jackson System Design (JSD) [32] is a variety of prescriptive practices, detailed views on how to use them, combine. With functional decomposition method combines an object-oriented design approach, and try to deal with most items in the software development life cycle - from analysis to implementation. A modelling stage, the network platform and an implementation phase are JSD three different stages.Stage modelling, graphical (tree-like) structure, process diagrams, analyzed and entities that affect the real world in which the actions (or events) in terms of the representation, are built. Network stage, a (graphical) system is a complete system in which the specification of a network of interconnected network of communicating processes described construct. Finally, the implementation stage, processes, network development, "the selection of a sequential run in a programming language is transformed.

C. Process Models

In order to manage the software engineering work, considerable attention has been focused on the development process. Already engineering disciplines to learn from the software community has developed its own process, or the life-cycle models. Especially in the stages of development of these models, deliverables, and verification and validation activities of the stresses.

Several steps are identified and the various tasks associated with these steps. For example, we typically a requirements-capture phase, specification phase, a design phase and out. Each of these phases, the phases of the input for the next phase inputs, technical and management tasks related to the stage, and is defined in terms of deliverables. Within each phase, development of methods and tools applicable to the description of deliverables to design, implement, test and is approved. Reviews and audits of the tools in the application phase, the end phase of the milestone reviews, etc. to ensure that the verification and validation is carried out. Reviewed the work produced within a step and based on that input, such as subsequent phases are placed under the control of the change so that it works.

D. Agile Modelling

Agile, using a series of short-term goals based on urgent priorities, provides value quickly. Architecture is based on the long-term objectives, using a set of basic principles, will increase the value of care. The two will be different, but the architect of the four active projects to bring them together at well-defined points: challenging issues within sprints through close cooperation, through the introduction of certain construction work at the time of the initiation of the project by setting the direction of the building, through storyboarding, and working software to be distributed by direct inspection. Tracking the implementation of the project, the construction of the state, advocating the merits of the development of architecture throughout, resurrected form of construction work, decomposing, and all active projects in a wide range of enterprise architecture, which led to the driving side: this should be the four key skills. When done effectively when the two with agility, this approach is workable balance between the business and the building achieves preferences.

Agile development processes and model-driven development (MDD) implementations has been adopted widely in the industry for years. The legacy of the advantages of agility and MDD, MDD agility introducing more and more into the field of education published the results of [12]. For example, in Ambler Agile, Extreme Programming into modelling and explains how to combine MDD. [12], Zhang better productivity, quality, and provides a method for the MDD apply to Extreme Programming techniques.According to the survey, as reported in a global developer, Agile Software Development (ASD) 35% gone mainstream and traditional software engineering at (TSE) 21% and the waterfall software development (WSD) 13% prevalent at [9].

E. Design Driven Modelling

Software system designed with a special regard to the software quality [13] [14] said in front of a number of key benefits for programming. Test Driven Development (TDD) is a well-known design and testing process for the detailed design level. But an examination of TDD technology, rather than in the written tests to the front of the product code is not a development and design technique. Gradually over the course of its run when it's tested and pass the test, and accordingly increase the efficiency of the internal structure of the code, the code is refactored added.

Design Driven Testing (DDT) defines that design should drive testing as opposed to tests driving design (TDD). Automated support for the product, which includes models from the unit tests, but also at the level of the scenario and the need to support acceptance testing. Unit testing catches "errors of commission", but not ""errors of omission", both the levels and the quality of the development of an efficient assessment [8], [21] is required.

Test automation is the focus of a great deal of research, however, manual testing is still the most widely used and appreciated by the software industry, and the future is likely to be replaced by automated testing [1]. TDD number of studies to analyze the development of the software and in some case the software architecture re-design studies, [3] [7] [16], but for reducing software defects in design-driven testing (DDT) method to demonstrate that there is no experimental study on improving software quality and price.

Design Driven Testing (DDT) that look back into the design of the unit test case skipping in favour of the design, but also through the effective up-front analysis and design to be able to solve most of the labour force and to reduce the cost [21][22]. DDT agile and test-driven approaches is the result of fusion with the up-front analysis and design. Therefore DDT in TDD tests, tests to verify that the design efficiently help reduce software defects, mainly around it, the tests are run, from design to drive the other way, and if used in the design of the drive.

V. Related Works

Traditional Software Development [24] organized in a hierarchical structure with multiple layers, is manager-led teams. The traditional command and control style of management teams, [17] usually is. The roles of the traditional teams, programmers who are responsible for such programs is reflected in the functional tasks of their organizational roles, which are based around, members of the team entrusted with the task of testing their managers, etc., are responsible for requirements analysis, analyst, responsible for the testers[4]. Achievements and the documentation of traditional teams, features, and planning [6], [17] there. There are indirect lines of communication across the different layers of the organizational hierarchy. The hierarchical structure of the members of the general empowerment of the entire project team was not visible [24].

Agile methods of the traditional software development models [18] have been developed in response to the weakness. Each iteration of Agile methods allows the user to focus on a small set of prioritized in the activity, the development of a methodology and incremental style of the traditional software development models, in particular by accommodating the changes. Agile method to encourage continuous customer interaction and feedback, and allow the customer the first priority is to develop the desired properties.

Scrum was developed by Jeff Sutherland and formalized by Ken Schwaber [5]. Scrum derives its roots from the documents. The scrum is usually from two to four weeks, the wheel will have to work on sprints. The customer with the highest value features first developed in such a way that at the time of each sprint, self-organizing teams choose to work from a prioritized list of customer needs. At the end of each sprint, a potentially shippable product is delivered.

eXtreme Programming (XP) was developed by Kent Beck and Ward Cunningham, with support from Ron Jeffries and Martin Fowler [23]. XP in the face of vague or rapidly changing requirements of the development of software for small to medium sized teams with a light weight system [23] is defined. XP to deal with small business values, and such schedule slips, cancelled projects, the inability to solve the business problem, and the richness of features, some of the classics, such as the development of the software has been developed to solve the problems.

John, bass and Kates then they are an integral part of their solutions for software architects, text-based recommendations for the Unified Modelling Language (UML) diagrams replaced [19], shifted slightly different approach. The decision of the industry and developers to test their early samples taken after realizing that seemed reluctant to accept ready-to-UML, and recommendations on how to design their systems very easily.

X Ferre et al. proposed architecture models including the use of software applications, [20] the possible construction of the proposed solutions. They are more comprehensive features that are specific solutions have been proposed in the literature on the use of well-known defined attributes spoils. The solutions to the experiment, which was compiled by the approval of the structural components, expressed in terms of. However, they still run directly by software developers are very abstract.

Rafiq Y et al. [2] The external code quality and productivity of Test-Driven Development (TDD) to investigate the effect of a systematic meta-analysis of 27 studies. The results, in general, TDD, but a small positive effect on productivity has had no discernible effect on the quality of the show. However, a sub-analysis of scholarly studies, quality improvement and productivity are too large compared to the drop in both industrial studies found. The big improvement in the quality of academic studies have shown that when the test was significant difference in the effort; However, due to the lack of data regarding the receipt of Industrial Studies. Finally, the developer experience and the impact of the amount of work as moderator variables, and a statistically significant positive correlation was found between the magnitude of the improvement in the quantity and quality of work.

VI. Conclusion

This paper presents a software quality engineering process for the improvement study and its possibility to predict the quality of a software process. It tries to present a review on this issues related to software engineering, designing, modelling and management practices.Most of the undeveloped design standards of most engineering disciplines, software developmenthave enormous complexity of the tasks need to be tackle. Seeking a systematic approach to control this complexity, the software industry needs for new methods and techniquesfor the improvisation system development.Design-based testing is relatively extensive software testing policy debate. Using automated test generation and execution, it from model sample for test automation expedition. Effective use of this approach to modelling skills such testing, but also as a good tool support programs and new skills, technology and knowledge. Internal or external quality and productivity of design and its impact on the quality of the build is a good topic for the future research.

References

[1]. Gordon Fraser and Andrea Arcuri",Whole Test Suite Generation",IEEE Transactions On Software Engineering, Vol. 39, No. 2, February 2013

[2]. YRafique and Vojislav B. Misic",The Effects of Test-Driven Development on External Quality and Productivity: A Meta-Analysis", IEEE Transactions On Software Engineering, Vol. 39, No. 6, June 2013

[3]. Gustavo Soares, RohitGheyi, and Tiago Massoni",Automated Behavioural Testing of Refactoring Engines", IEEE Transactions On Software Engineering, Vol. 39, No. 2, February 2013

[4]. J Itkonen, M V. Mantyla and C Lassenius, "The Role of the Tester's Knowledge in Exploratory Software Testing", IEEE Transactions On Software Engineering, Vol. 39, No. 5, May 2013

[5]. Scrum Alliance, World Wide Web electronic publication, www.scrumalliance.org/view/scrum_framework, Sept. 2008.

[6]. S. Nerur and V. Balijepally, "Theoretical Reflections on Agile Development Methodologies", Comm. ACM, vol. 50, no. 3, pp. 7983, 2007

[7]. NorakmarArbainSulaiman, MurizahKassim, "Developing A Customized Software Engineering Testing For Shared Banking Services (SBS) System",IEEE International Conference on System Engineering and Technology,2011

[8]. Mohammad Wahid, Dr Abdullah Almalaise",JUnitFramework: An Interactive Approach for Basic Unit Testing Learning in Software Engineering",International Congress on Engineering Education,2011

[9]. D. West, T. Grant, M. Gerush and D. D'Silva, Agile development: Mainstream adoption has changed agility, Forrester Research, 2010.

[10]. Forrest Shull, Carolyn Seaman, and Marvin Zelkowitz, "Victor R. Basili's Contributions to Software Quality", IEEE Software Jnl, pp. 1618, Jun 2006.

[11]. Watts S. Humphrey, "PSP: A Self-Improvement Process for Software Engineers", SEI Series in Software Engineering, Addison-Wesley, 2005

[12]. Yuefeng Zhang, "Test-driven modelling for model driven development", IEEE Software, v21, Issue 5, September, 2004

[13]. L. Bass, P. Clements, and R. Kazman, "Software Architecture in Practice", 2nd Edition Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003.

[14]. David L. Parnas,MarkLawford, "Inspection's Role in Software Quality Assurance", IEEE Software Published by the IEEE Computer Society, 2003

[15]. Matt Stephens and Doug Rosenberg, "Design Driven Testing", Apress, 2010.

[16]. Muhammad Ali Babar, Barbara Kitchenham, "Assessment of a framework for comparing software architecture analysis methods", in Proc. 11th International Conference on Evaluation and Assessment in Software Engineering, Keele, England. Citeseer, 2007.

[17]. S. Nerur, R. Mahapatra, and G. Mangalaraj, "Challenges of Migrating to Agile Methodologies", Comm. ACM, vol. 48, no. 5, pp. 72-78, 2005

[18]. D. Robey, R. Welke, and D. Turk, "Traditional, Iterative, and Component-Based Development: A Social Analysis of Software Development Paradigms", Information Technology and Management, vol. 2, no. 1, pp. 53-70, 2001

[19]. B. John, L. Bass, E. Golden, P. Stoll. "A Responsibility-Based Pattern Language for Usability-Supporting Architectural Patterns". Proceedings of the 1st ACM SIGCHI symposium on engineering interactive computing systems, 2009.

[20]. X. Ferre, N. Juristo, A.M. Moreno, M. Sánchez-Segura. "A software architectural view of usability patterns". Proceedings of INTERACT Conference. 2003.

[21]. A. Monden, T.Hayashi and et.al, "Assessing the Cost Effectiveness of Fault Prediction in Acceptance Testing", IEEE Transactions On Software Engineering, 2013

[22]. CemalYilmaz",Test Case-Aware Combinatorial Interaction Testing", IEEE Transactions On Software Engineering, Vol. 39, No. 5, May 2013

[23]. K. Beck, Extreme Programming Explained: Embrace Change, first ed. Addison-Wesley Professional, 1999.

[24]. J.E. Tomayko and O. Hazzan, Human Aspects of Software Engineering. Charles River Media, 2004.
[25]. Roger S. Pressman, "Software Engineering: A Practitioner's Approach", 4th Edition, McGraw Hill Inter. Editions, 1997.
[26]. Alford, M. W., "SREM at the Age of Eight: The Distributed Computing Design System", Computer, IEEE Computer Society Press, 18(4): 36-46, April 1985.
[27]. Yourdon, E. and L. L. Constantine, "Structured Design", Prentice-Hall International, Englewood Cliffs, New Jersey, USA. 1979.
[28]. Ashworth, C. and M. Goodland, "SSADM: A Practical Approach", McGraw-Hill Book Company Europe, Maidenhead, UK, 1990.
[29]. Ross, D. T. and K. E. Schoman, "Structured Analysis for Requirements Definition", Transactions on Software Engineering, IEEE Computer Society Press, 3(1):6-15, January 1977.
[30]. Mullery, G.",Acquisition - Environment"; (In) Distributed Systems: Methods and Tools for Specification; M. Paul and H. Siegert (Eds.); LNCS, 190, Springer Verlag, 1985.
[31]. Kramer, J. and A. Finkelstein, "A Configurable Framework for Method and Tool Integration"; Proceedings of European Symposium on Software development Environments and CASE Technology, Germany, Springer-Verlag.June 1991.
[32]. Jackson, M. and P. Zave, "Domain Descriptions"; Proceedings of International Symposium on Requirements Engineering, Coronado Island, San Diego, 56-64, IEEE Computer Society Press, USA, 4-6th January 1993.