International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)
Web Site: www.ijettcs.org Email: editor@ijettcs.org
Volume 5, Issue 2, March - April 2016 ISSN 2278-6856

A Systematic Analysis on the Components of Software Process Layered Architecture

Jogannagari Malla Reddy[1], S.V.A.V. Prasad[2] , Sambasiva Rao Baragada[3]

[1]Department of Computer Science and Engineering, Lingaya's University
Faridabad, Haryana State, India

[2]Lingaya's University, Faridabad, Haryana State, India

[3]Department of Computer Science, Government Degree College,
Khairatabad, Hyderabad, Telangana State, India

Abstract
The software engineering intended to assist for the the development of software products. Software product is a logical and knowledge attribute compared to the other products in nature. The knowledge area renamed the software engineering environment into software layered technology which plays key role in developing software product. The layered technology consists of various layers like quality, process, methods and tools. Quality is defined as making the software product to satisfy all quality attributes achieved by the stakeholder. The quality factor must be involved in the whole process of software development. The software developers attempt to understand software processes, methods and tools. The purpose of this paper is to analyze and assess layers influence the extent usage in improving the quality of the software.

Keywords: Software Process layer, Quality Focus, Process, Methods, Tools

1. INTRODUCTION

The software engineering is progressed within a short span of time compared to the classical engineering like civil, electrical and mechanical etc. The software industry is too young compared to other industry. Today, software is nucleus for other industries to making world as digital in building the monstrous systems, in terms of size and complexity. The nature of complexity is increased in the last 40 years. In 1970 the application software executed on uniprocessor generated with alphanumeric output, and received input from a linear source. Now the applications are complex executed on multi processors under the graphical user interface with client-server technology. The information disseminated geographically with distributed systems.
The Software industry is evolving continuously and developing enhanced products to fulfil the ever-changing business requirements. The endless maintenance however, decreases the internal quality of the software over time. This software erosion phenomenon is resulted in increased testing effort and operational expenditure.
The software engineering activity is complex which involves interactions between people, processes and tools for the software development. The software engineering is a layered technology which consists of process, methods and tools which are targeted with bedrock of quality focus. The quality of software has required in most of the industrial sectors. The objective of software engineering is to develop the quality software at on time, with in the budget. The observer views the software quality is in different views. It is difficult to define which contain the range of attributes and varies according the viewpoint of the observer. The software quality standards are needed to achieve acceptable levels in both software products and process. The process is important because industry cares about their intrinsic qualities, such as uniformity of performance across different projects. The software industry has been seriously looking for effective methods to improve the quality products to satisfy the customers. Tools provide additional support in process and methods development.
This paper has reported out empirical study of software layered technology from the perspective of software quality. The section 2 explains the related work and taxonomy of software engineering. The section 3 consists of general definitions of software engineering and its objectives. Section 4 presents overview spectrum of the software layered technology and finally concluded with discussion in section 5.

2. RELATED WORK

The software engineering in the early years is dominated by the idea of structure, both in terms of the product and process. A number of theorists and researchers have worked on software engineering issues in the domain of quality, process, methods and tools. The developed taxonomy can be benefit to extend the knowledge in software engineering for discussion, setting up and evaluating the more quality software product within the time schedule and budget.
In 1980, the basic principles and Concepts for achieving Quality for software framework developed for Department of Defense in the context of Capability Maturity Model Integration.
Paulk et al. 1993 defined the intent of software process improvement is improving software product quality,

increasing productivity, and reducing the cycle time for product development.
David Carrington [1] provides the breakdown of topics within the software engineering infrastructure knowledge area. The author discussed more details on software engineering methods and tools.

Ron Burback [2] in the year 1998 presented a thesis on nature of software engineering cyclical methodology and progressive sequential methodology at Stanford University.
Forselius and Kokola [3] proposed the research on software Project Estimation and Measurement Systems.
Maria et al. [4] have pursued a research to re evaluate the first widely accepted software quality model derived by Bohem et al [5].
The organizations adapted ISO stands of quality in software development to excel their performance. ISO/ICE 9126 quality model have various internal and external quality factors [6].
Gerhard Weiss et al [7] presented the chapter on software process and tools in practical way.
Shahid Iqbal et al. [8] highlighted metrics, skills and processes which can help project managers and stakeholders for managing and monitoring the IT projects effectively.
Alan Devis et al. [9] presented a paper on software requirement specification identification and measuring its quality. The paper thoroughly the concept of quality in SRS and defines attributes and suggested the techniques for measuring.
Mohammad Ubaidullah Bokhari et al. [10] discussed the requirement analysis with metrics and usage of automated requirement tools, which will useful in software development.
Mohd Haleem et al. [11] elaborated the quality metrics in requirement engineering process of software development life cycle.
Mohd Ehmer Khan and Farmeena Khan [12] have discussed the role of testing in the various phases of software development life cycle.
Jeng-Nan Juang and S. Radharamanan [13] determined the significant factors and their effects on software engineering process quality. Their paper is useful for project managers to monitor the process performance and act according to any abnormalities.
Pankaj Jalote in one of his publications mentioned that a high maturity organization is expected to use metrics heavily for project process management.
Kitchenham [14] defines quality as “Quality is a complex concept”. Different stakeholders have different views on software quality. There can be no single, simple measure of software quality acceptable to everyone. It is highly context dependent.
R. Fitzpatrick [15] expressed the software quality as an extension of industry defined set of desirable features, are incorporated into a product to improve its performance.

3. GENERAL DEFINITIONS OF SOFTWARE ENGINEERING & ITS OBJECTVIES

The conference sponsored by the NATO Science Committee held in Europe in 1960 is focused to discuss on the crisis of software development. The committee initially proposed the software engineering landscape. The software engineering landscape could be perceived as a collection of dominant features of workable frameworks of planning, conceptualizing and design of software. The subject concerns with well defined principles, methods, models, standards and procedures which makes possible to manage, plan, model, design, implement and evaluate the software system.
This section looks into various definitions of software engineering defined by eminent theorists and reputed organizations by an overview of its feature.

- The software engineering as establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines”.- [Fritz Bauer]
- The software engineering is the application of science and mathematics by which the capabilities of computer equipment are made useful to man via computer programs, procedures, and associated documentation. [Boehm].
- Software engineering is the science and art of specifying, designing, implementing, evaluating with economy, timeliness, elegance programming and documentation procedures where by computers can be made useful to man. [McDermid] [4] .
- The software engineering is “knowledge or eliminatory rules that constitute the foundation and essence of software engineering”. [Guay 2004].
- The Software engineering as application of systematic, disciplined, quantifiable approach to the development, operation and maintenance of software (IEEE).

The software Engineering goals are to develop the quality software products with in the stipulated time and budget
Cost of developing the system is the cost of the resources used for the system, which, in the case of software, hardware, manpower and other support resources. Software development is knowledge intensive. The cost of software project is measured in terms of person-months, i.e the cost is considered to be the total number of persons spent in the project. To convert this dollar amount, it is multiplied with the dollar cost per person-month. In defining this unit cost for a person-month, the other costs are included. In this manner, by using person-months for specifying cost, the entire cost can be modeled.
The schedule is an important factor is to develop the software product with required features in stipulated time. The business rends requires that time to market of a product should be reduced; i.e the cycle time from concept to delivery should be small. The schedule is depends on the project complexity, intellectuality of manpower in terms of programmer productivity and project size. The

project time can be reduced with rapid application development and highly sophisticated tools.
One of the major factors driving, the software development is quality. In the recent business trends the quality is the main "mantra". The cost of product is based on the quality. Clearly, a set of innovative methods that can produce the high-quality software is another fundamental goal of software engineering.
Various theories define quality as:

- "Software Quality is the degree to which a system, component, or process meets specified requirements and meets customer needs or expectations". [IEEE]
- "Quality is the totality of features and characteristics of give needs". [ANSI Standard 1978]
- "You cannot control what you cannot measure". [Tom Demarco]
- "I know no way of judging the future, but by the past". [Patrick's Henry].

Lack of conformance to requirements is lack of quality.

4. THE SOFTWARE LAYERED TECHNOLOGY

The software layered technology of software engineering consists of process, the management technical methods and use of tools to develop the software products. The aim of software engineering is committed to quality. Software layered technology is classified into quality focus layer, process layer, method layer and tools layer. The layers are related and each layer demands the fulfilment of previous layer. Figure 1 depicts the upward flowchart of the layers of software development.

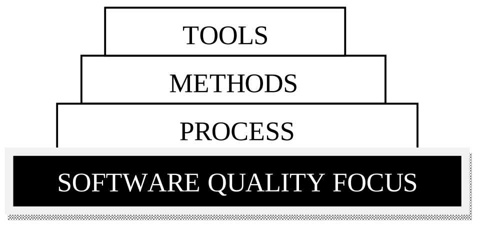

Figure 1 Software Layered Technology

4.1 Quality focus layer

Quality focus is the bedrock of software engineering. An attribute of software that contributes to its quality, where quality is the degree to which software meets customer or user needs or expectations [IEE90]. For example, maintainability, an attribute that contributes to quality, is a factor.
Quality factors are stake holders originated attributes that users expect to see in the delivered software product. Metrics are developer oriented units that estimates the quality. In validating metrics an attempt is made to identify a statistical relationship between metrics and quality factors. If validation is successful, it is needed to use the validated metrics as estimates of quality during software design. The relationship between a quality factor and metrics allows controlling the quality of software projects. Various philosophies like total quality management, six sigma, CMMI and statistical analytical process are targeted to improve the quality culture. Various quality models plays important role for improving the quality of software product. Quality models like Mc Call's, Boehm, Dromey and FURPS are available to fulfill the objective of getting quality. Each model has its own characteristics and guides the project team to improve and maintain the software quality.

4.2 Process layer

The process layer acts as a foundation of software engineering. The process layer defines a frame work comprised of a set of key process areas (KPAs) for management control and effective delivery of software projects. This establishes the context in which technical methods are applied work products such as models, documents, data reports, forms etc are produced. The various tasks can be performed in this layer.

- Determining Deliverables
- Establishing milestones
- Software configuration /
- Change management.
- Software Quality Assurance

Software Engineering process is the glue that holds all the technology layers together and enables the timely development of software product. A structured set of activities required to develop a software system. Many different software processes but all have specification, design, coding & validation, implementation and evaluation.

The process descriptions may consist of the following:

Products:	The outcome of a process.
Roles :	The responsibilities of the people involved in the process.
Conditions:	Associated with a statement that is true before and after the process activity.

The process activities are classified into plan-driven and agile process. The plan driven process, planned in advance and progress traced against the plan. The agile process, the planning is incremental and easier to change as per customer requirements. In practical sense, most practical processes include elements of plan driven and agile process. There is no way of right or wrong software process to adopt. The software process model is an abstract representation of process on some particular perspective. Various processes models are used to develop the software systems. Each model encompasses advantages and disadvantages. The selection of process model is based on nature of project and usability.
Waterfall model: This model is a linear sequential, starts with customer requirements, progresses with design, construction and ended with implementation and evolution.

Advantages & disadvantages:

- Waterfall model is the oldest paradigm of sequential classical life cycle with systematic approach.
- It is difficult for the customer to state all requirements explicitly. Requirements are not revealed initially, it is problematic to accommodate in mean time.
- The stake holder should have patience. A working version of the program will not be available until late in the project time span.
- Blocking states. – (Some of the team members
- should wait for tasks of other members should complete. The Real time projects rarely follow.
- However, it can serve as a useful process model in situations where requirements are fixed and work is to proceed to completion in a linear manner.

Incremental Model*:* This model represents the same phases of water fall model but in the successive increments.

Advantages:

- Incremental model combine the features of water fall model in iteration fashion applies linear sequence in staggered fashion in calendar time progress.
- Each linear sequence develops a deliverable increment. The first increment is core of the product. Successive increments are supplementary features. The progress of model continues until completion of production.
- Incremental process model is useful when staffing is unavailable for complete implementation by the business deadline that has been established for the project. Early increment can implemented with less people. If the core product is well received, additional staff can be added to implement the next increment. Increments can be planned to manage the technical risks.

Rapid Application Development (RAD) Model : This model is high speed adoption of water fall model that emphasizes a short development cycle. RAD could be achieved through component based construction approach. If the project requirements are well understood in prior, RAD process enables a development team to create a fully functional system within short period of time. Planning is made for multiple software teams which work in parallel on different aspects. RAD modeling encompasses on three major phases namely business modeling, data modeling and process modeling. Each major function can be address by separate RAD teams and then integrated as whole.

Advantages & disadvantages:

- Development time is 60 to 90 days.
- For large scale projects RAD requires more man power to form right number of RAD teams.
- If developers & customers are not committed then the situation might be problematic to complete within the time frame.
- If the system cannot be properly modularized, building the components necessary for RAD will be complicated.
- Performance is to be achieved through tuning the interfaces to system components and in such a case RAD approach may not work.
- RAD may not be appropriate when technical risks are high.

Prototype Model : This model is not based on strict planning, but it is early approximation of the final software product. A prototype acts as a sample to test the process which we learn and try to build a better final product. The prototype may or may not be completely different from the final system we are trying to develop

Advantages & disadvantages:

- The direct involvement of the users in development will get the better understanding of the system being developed.
- Errors can be detected early with feedback for better solutions. Missing and confusion functionality can be identified easily.
- The more involvement of the client is not always preferred by the developer. It will disturb the rhythm of software development.

Evolutionary Development: This approach could be perceived as a set of interleaving the activities and specifications in development and validation. The development of system starts at initial stage with abstract specification, then refined with customer input to produce new refinement in repetitive which satisfies the customer's needs.

Spiral Model : The Spiral model is proposed by Boehm, couples the iterative nature of prototyping with features of water fall model. The Evolutionary process begins the software development teams to perform activities that are implied by circuit around the spiral clock wise direction begins at centre. Spiral model is decomposed into set of framework of activities. Each framework activity represents a segment of the spiral path. The first circuit around the spiral represents a concept development which starts at core of the spiral and continuous for multiple iterations until the customer satisfaction.

Features:

- The Spiral model is systematic stepwise approach suggested by waterfall model, but incorporates into a repetitive frame and more realistic for real world applications.
- .- It provides the potential for rapid development, yields more complete versions of software. Each version is treated as risk reduction.
- This model is treated as a realistic process model for the large-scale software systems.

Apart from the above process models, software process management manages the effects of change throughout the software process and the software quality assurance conducts the activities required to ensure the software quality.

4.3 Methods layer

Software engineering methods provides the technical knowledge of how to build the software systems. Methods provide notation and vocabulary, procedures for performing identifiable tasks and guidelines for checking both the process and the product. The development methods widely in scope from single life cycle phase to the complete life cycle. Methods encompass the array of tasks of the following:

Communication: This framework activity involves communication with customer and other stake holders for gathering requirements.

Requirement Engineering : This task provides appropriate mechanism for understanding what the customer wants and its need, assessing the feasibility, negotiating a reasonable solution with more unambiguously, validating specification and transforming into operations system. Requirement engineering process accomplishes various tasks like inception, elicitation, elaboration, negotiation, specification, and validation. Requirements collected with various methods, sample surveys, questionnaires, personal interviews, group discussions, sceneries and ethnography.

Software Design : Design consists of a set of principles, concepts, and practices to develop the high quality software product. Software products developed with classical engineering failures due to the lack of depth in modeling, flexibility and quantitative nature. However recent software products designed with more quality, depth and quantitatively measured. Design encompasses the representation of models for better understanding of software requirements. Modeling is a diagrammatical representation of proposed system. Designing a system include developing of data flow diagrams (DFDs), grid charts, flowcharts, connectivity diagrams, decision trees and decision tables for a clear representation of the proposed system. Design model or representation exhibits the firmness, delight and commodity of system's nature.

Construction and Testing : This activity combines code construction (either manual or automated), debugging and testing process to maintain the software quality. The objective of this phase is to develop the suitable software code for required functionality. Various models designed during inception and elaboration phases of unified process are considered for code generation. Developed code can further be refined using debugging to eliminate software errors. Testing starts with unit testing, progresses with integration, system testing and concludes with acceptance testing.

Deployment and Evolution : Deployment deals with the implementation of software product in the real time environment. Software product could be implemented using several methods like direct approach, phased approach and parallel approach in the real time. Evaluation represents the collection of feedback from the stake holders and maintains the software with the intent of customer satisfaction. Software needs to be evaluated with process changes and technological changes. Evaluation process conducts audit trials, collects feedback, generates daily reports and maintains system logs. Poor maintenance of software leads to stakeholder un satisfaction.

The generic view of framework describes the number of umbrella activities namely risk management, software quality assurance, software project tracking and control, formal technical reviews and configuration management embedded with software engineering actions. Each action highlighted with work tasks that accomplish some part of the work.

4.4 Tools layer

Automated and semi-automated tools provide support for pursuing process and methods. Automated tools reduce the cognitive load of the software engineer. Tools are intended to make development more systematic, vary in scope from supporting individual tasks to encompassing the complete life cycle. Tools which are integrated for software development are called Computer Aided Software Engineering (CASE) tools. CASE tools may include editors, databases, test case generators and code generators which automatically generates the source code from the system models.

Methods that correspond to the software development life cycle provide a location for specific tools as following:

Software Requirement Tools : These tools help in classification of modeling and traceability based on requirement literature.

- Requirement modeling tools: Tools used for eliciting, recording, analyzing and validating software requirements.
- Traceability tools: These tools help in understanding the complexity of software systems to trace the stakeholders' requirements.

Software Design Tools : Design tools used to create and evaluating software design. A variety of tools perform diversified design notations and methods. Manual techniques like top down design, bottom-up and structural paradigm could also be employed along with usage of automated tools during design. Design expression and configuration aid, process design language (PDL), higher order software are some automated design tools provide design analysis and consistency checking features.

Software Construction Tools : Construction tools are concerned with the development of source code to enable machine execution.

- Program editors: Editors are tools used to create and modification of software programs. These are general purpose text or document editors and are computer language specific. Normally, editors are human-controlled development tools.

- Compliers and code generators: These are non-interactive translators of source code. An integrated programming environment provides integration of compile and program editing facilities. These include pre-processors, linker/loaders and code generators.

- Interpreters: These tools provide the execution of software through emulation. These tools support software construction activities with controllable and observable environment for program execution.

- Debuggers: Debugging tools support the construction process but are different from program editors or compliers.

Software Testing Tools : Software testing tools classified into the following:

- Test generators: Test generators assist the development of test cases.

- Test execution frameworks: Test execution frameworks enable the execution of test cases, where the behaviour of the test object is observable. .

- Test evaluation tools: Test evaluation tools support the assessment of the results of test execution to determine the observed results conforms to the expected results.

- Test performance analysis tools: These tools are intended for measuring and analysing software performance. These are the specialized tools of testing where the goal is to assess the performance behaviour of software rather than functionality.

Software Maintenance Tools : These tools are intended for software maintenance where an existing system is being modified. Software maintenance tools are further decomposed into comprehension and re-engineering tools.

- Comprehension tools: The tools include visualization tools such as animators and program slicers.

- Re-engineering tools: These tools allows to translation of program to new programming language, or dataset to a new format. Reverse engineering tools assist the process by working backwards from an existing product to create abstract artifacts to generate a new product from an old one.

5. CONCLUSIONS AND DISCUSSION

Software engineering involves several expertise stages right from defining the problem, analysis, design, coding, testing and maintenance. Numerous advanced process models, tools and methods have been developed to enable the software engineers in developing user friendly software products. Software engineering can be defined as a systematic approach of software development which is concerned with the aspects of a qualified software production. Estimation of project size and cost, planning, tracking, formal technical reviews, bug inspections, configuration management, quality assurance and risk management are the key concepts followed during software engineering. This paper analyses the significance of layers with the intention of improving the quality of the software. This systematic analysis could drive in evaluation of software process, methods during development of a software project. The work can be further used as a base in inter-comparison of two or more software process layers.

References

[1]. David Carrington, “Software Engineering Tools and Methods”, IEEE- Trial Version 1.00, May 2001.
[2]. Ron Burback, “Software Engineering Methodology: The Watersluice”, Doctoral Thesis, Stanford University, 1998.
[3]. Forselius.p and Kakola.T, “An Information Systems Design Product Theory for Software Project Estimation and Measurement Systems”, 42nd Hawaii International Conference, Jan 2009.
[4]. Maria Sverstuk et al, “Software Quality: What is really important and Who Says So”, International Conference NIMESTIC 2000, 11-13 Sep 2000.
[5]. Boehm BW, Brown J.R, Lipow M, “Quantitative Evaluation of Software Quality”, Proceedings of Second International Conference of Software Engineering, pp 592-605, 1976.
[6]. ISO “ISO 8602”, 1994.
[7]. Gerhard Weiss, “Procedings of the famous 1968 and 1969 NATO Software Engineering Workshops”,
[8]. Shahid Iqbal et al, “Yet another Set of Requirement Metrics for Software Projects”, International Journal of Software Engineering and its Applications, Vol. 6, Jan 2012.
[9]. Alan Devis , “Identifying and Measuring Quality in a Software Requirements Specification”.
[10]. Mohammad Ubaidullah Bokhari1 and Shams Tabrez Siddiqui, “Metrics for Requirements Engineering and Automated Requirements Tools”, Proceedings of the 5th National Conference INDIACom-2011, Mar 2011.
[11]. Mohd. Haleem et al, “Overview of Impact of Requirement Metrics in Software Development Environment”, International Journal of Advanced Research in Computer Engineering & Technology, Vol. 2, No. 5, May 2013
[12]. Mohd, Ehmer Khan and Farmeena Khan, “Importance of Software Testing in Software Development Life Cyle”, International Journal of Computer Science Isues, Vol. 11, Issue 2, No 2, March 2014.
[13]. Jeng-Nan Juang and R. Radharamanan, “Determining Significant Factors and their effects on Software Engineering Process Quality”, ASSEE Publications, Fall – 2009.

[14]. Barbara Kitchenham and Shari Lawarence Pfleeger, "Software Quality: The Exclusive Target", IEEE publication, Jan 1966.
[15]. R. Fitzpatick, "Software Quality: Definitions and Strategic Issues", In School of Computer Report, Country of Staffordshire,UK, Staffordshire University, 1996.
[16]. R.S. Pressman, "Software Engineering – A Practitioner's Approach (4th Ed), McGraw-Hill.

AUTHOR

Jogannagari Malla Reddy obtained M.Tech(CSE) from JNTU, Hyderabad. and pursuing Ph.D(CSE) from Lingaya's University, Faridabad. His area of specialization in software Engineering, Management Information Systems and Database Management Systems. He published various papers on Information Systems in reputed National and International Journals & conferences.

Dr. S.V.A.V. Prasad done his PhD from Andhra University, presently working as Professor & Dean(R&D) in Lingaya's University, Faridabad, has 30 years of experience in Teaching and R&D. He published various research papers in National and International reputed Journals & Conferences. He guided many students in research progrmmes at university in communication system and information system areas.

SambasivaRao Baragada received his M.Sc and M.Phil in Computer Science in 2001 and 2006 respectively. He is awarded his Ph.D in Computer Science from Sri Venkateswara University, Tirupati in the year 2011. Earlier he was associated as Scientist at Satellite Data Acquisition & Processing System (SDAPS), Data and Information Management Group (DMG), Indian National Center for Ocean Information Services (INCOIS), Ministry of Earth Sciences, Govt. of India, Hyderabad. At present he is working as Assistant Professor in Computer Science and Head, Dept.of Computer Science, Government Degree College**.**