DESIGN AND DEVELOP A SOFTWARE REQUIREMENT IDENTIFICATION TOOL

Sudhir K. Singh*
Dr. Raj Kumar*

ABSTRACT

Problem identification focuses on gaining an understanding of the customer's problem domain and identification of the root of the symptoms being observed by customer. Problem decomposition on the other hand is the process of translating our understanding of that problem into a statement of needs that provides the basis for solution specification. Software development activities performed by humans requires methodological shift to the description of what computers are required to solve as opposed to how computer are to solve a particular problem. Software development is highly dependent on the use of tools. These tools support and automate activities performed in different sub-domains of software development. The software development process definitions are used as inputs to create process models and provide actual implementations. The paper includes case-study work to identify the significance of the problems and the applicability of the method as a solution to issues in tool integration.

Keywords: CORE- Controlled Requirements Expression, IBIS- Issue-based Information Systems requirement analysis, RGM-requirement generation model,win-win spiral model

Introduction

The subject of software requirements is often given far less attention in software engineering education than software design, even though its importance is widely recognized. F.P. Brooks ponder over this dilemma in flowing words:

"The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all the interfaces to people, to machines, and to other software systems No part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later" [Brooks, 1987**]**

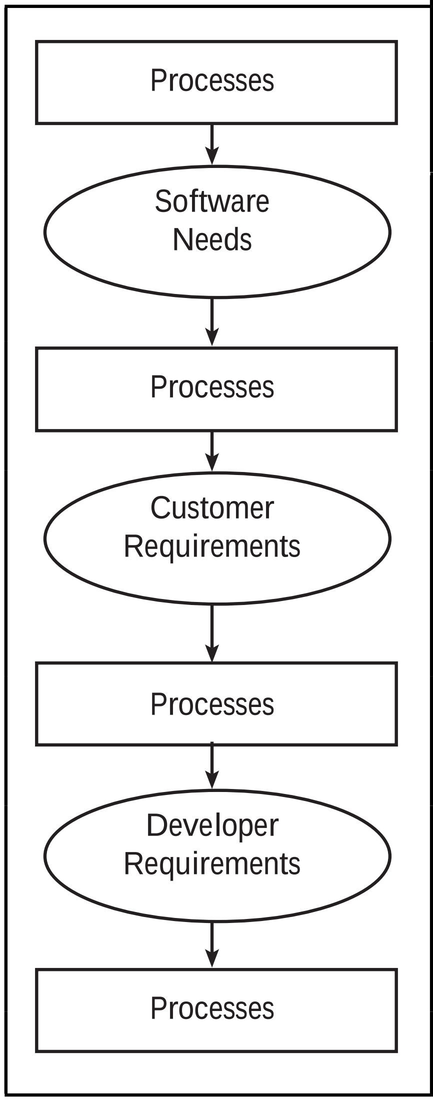
Early experience in building software systems showed that existing methodologies were inept. Major projects were, at times, years late. Those cost much more than originally predicted, were unreliable, difficult to maintain and performed inefficiently. The inadvertent delays and significant budget overruns caused software costs to rise, while hardware costs were plummeting. New techniques and methods were needed to control the complexities inherent in large software systems.

***Sudhir Kumar Singh** is a Research Scholar at Mewar University, Gangrar Chittorgarh, Rajasthan.

**** Dr. Raj Kumar** is the Director of Dr Rizvi College of Engineering, Kaushambi, Allahabad.

Figure: 1

Life-Cycle terminology used in this Tool



Existing Life-cycle Terminology	Terminology Used in this Tool
Market Analysis System Analysis Business Planning System Engineering	Context Analysis
Market Needs Business Needs Demands System Requirements	Needs Product
Requirements Analysis Requirements Definition System Specification	Customer Requirements Process
Requirements Requirements Definition Requirements Document Requirements Specification Functional Specification	Customer Requirement Product
Specification	Developer Requirement Process
Behavioral Specification System Specification Functional Specification Specification Document Requirement Specification	Developer Requirement Product
Design	Design Process

After more than 30 years of development, the software industry has made enormous progress in developing reliable software, more often than not, within budgetary and schedule constraints. We now have a better understanding of various processes involved in software development, and have modeled these processes in the form of software development lifecycles, lending much-needed structure to the overall development process. Nevertheless, many large software projects are still late and over budget. Moreover, software that is delivered often does not meet the real needs of the customer.

A requirement is something, which is desired or needed. From the software engineering point of view, a requirement is a condition or capability to which a system or software must conform to or a quality that the system must have. Requirements can be divided into two general categories, functional and nonfunctional requirements.

Functional requirements define the capability and the behavior of the system. They can be thought of as things that the system does on behalf of the user. Functional requirements are used to express the behavior of a system by specifying both the input conditions and the output conditions that are expected to result.

Non-functional requirements encompass a wide variety of attributes that do not specifically relate to the system's functionality. These kinds of attributes are, for example, quality attributes such as usability, reliability, performance and supportability. Non-functional requirements are usually as important to the end-user community as are the functional requirements.

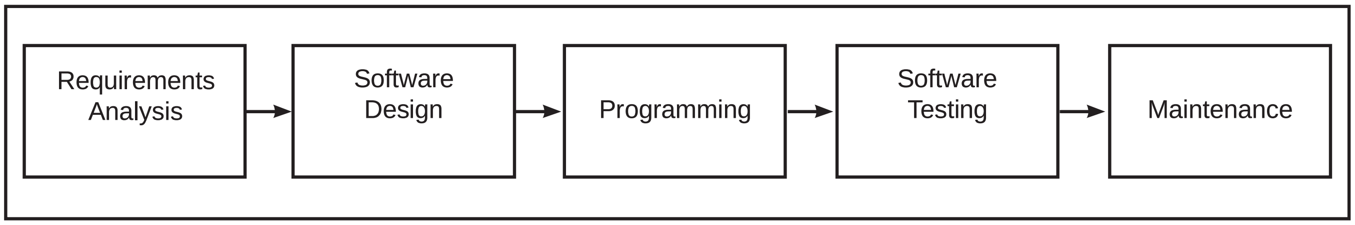
Requirements engineering can be split into two main areas of activities, requirements development and requirements management. Requirements development concerns activities related to elicitation, analysis, documentation and validation of requirements. It deals principally with the content of the requirements. The purpose of requirements development is to produce and analyze customer, product, and product-component requirements.

Requirements management concerns activities related to controlling and tracking of changes to agreed requirements, relationships between requirements and dependencies between the requirements specifications and between specifications and other project artifacts. Requirements management can be seen as a supportive process, which helps to manage requirements through the product's lifecycle.

2. Software Development Paradigms: A Software Development Paradigm outlines a development strategy that encapsulates processes, methods, artifacts and tools. Software paradigms typically consisted of five activities: Requirements Analysis, Design, Implementation, Integration, Testing and Maintenance. Different Software Process Models decompose these generic activities in different ways. The timing of the activities varies, as does the outcome.

Figure: 2

Waterfall Model



The first development model, the Waterfall Model offered means of organizing the development process. This model placed the five activities in a linear timeframe, with each phase following the previous one.

Software development has thus become a complex sequence of information transformations, with a pre-defined aim and several levels of input and knowledge. These information transformations create outputs that are used as inputs in succeeding steps towards the goal. These transformations are specialized into separate processes like requirements engineering, software design, coding, and software testing. As in any engineering domain, tools have been developed to support software engineers by increasing the efficiency of the execution of processes.

3. Requirements Generation Process: The Requirements Generation process is placed at the beginning in most software development lifecycles. In itself, the process is composed of the activities listed below.

- **Software Requirements** are elicited from people or derived from system requirements. An important precursor to the elicitation process is the Problem Synthesis process, during which the underlying issues are diagnosed and customer needs are elicited.

- **Requirements Analysis** is usually performed before the customer commits to the actual development process. The customer assesses the acceptable level of risk regarding the completeness, correctness, and technical and cost feasibility.
- **Requirements Specification** elicited and analyzed in the preceding phases are documented and expressed in the form of a formal document, often referred to as the Software Requirements Specification (SRS).
- **Requirements Verification and Validation** ensures that the requirements elicited and specified in the SRS adhere to the customer needs or the high level requirements. The requirements elicited are presented to diverse audiences for review and approval. Adherence of specified requirements to pre-defined quality attributes is also tested.
- **Requirements Management** must incorporate a management methodology that pervades through the entire process and facilitates communication effectively, in case any changes are made. A process that establishes and maintains agreement between the customer and the project team on the changing requirements of the system.

4. Requirements Generation Models: We provide a survey of existing requirements generation approaches, describing existing methodologies, models, frameworks and techniques.

- **CORE- Controlled Requirements Expression:** CORE is a prescriptive, top-down method. It identifies clients and their viewpoints, and describes dynamic aspects of the software system with the primary aim to determine functional requirements. CORE assumes that different viewpoints are necessary to specify requirements. Viewpoints are the set of interests to be supported or influenced by the proposed system. Example viewpoints are environment view, process view and reliability view. CORE defines responsibilities for clients and requirements analysts and structures their communication. The customer is responsible to resolve conflicting views. The requirements analyst is responsible for the documentation of requirements specifications, which express (multiple) viewpoints, and ensure their successful combination into a coherent specification of the software system. CORE proceeds iteratively, first defining the problem, then decomposing it into viewpoints, and acquiring information about each viewpoint. This recurrent process leads to a hierarchy of viewpoints. The RE process comprises viewpoint structuring, tabular collection, data structuring, single and combined viewpoint modeling, and constraint analysis.
- **IBIS- Issue-based Information Systems:** IBIS assumes that RE is fundamentally a conversation among clients and requirements analysts in which they pool their respective knowledge and viewpoints to specify the software system. It organizes and tracks the rationale underlying requirements. The method helps to point out where alternatives have not been considered and where specifications are strong and weak, based on supporting or objecting arguments. IBIS acquires requirements by keeping track of issues being discussed, propositions on these issues, and arguments in support of a position or object to it. A typical IBIS method scenario would be a stakeholder posting an issue containing a question such as: How should we do X? Another stakeholder may respond by proposing one way to accomplish X and can also post arguments supporting the proposal. Other stakeholders can post competing arguments or support the proposal. Hence the goal of such discussion is to understand each other's proposals and to persuade others aboutone's viewpoint.
- **Jackson System Development:** Jackson System Development (JSD) method comprises software development starting from problem analysis and ending with code. The method does not subscribe to traditional life cycle models. Instead of top-down, it develops functional specification middle-out

from an initial set of requirements. JSD follows five steps to specify requirements: entity-action & entity structuring, initial modeling, function step, information function step and system timing. The first two steps are concerned with creating a model of the application domain, the others with the specification of functions to be added to the model. The model of the application domain is created in terms of actions and entities. These are documented and constraints on time ordering of actions are expressed as structure diagrams. Initial modeling produces a model of the application domain in terms of communicating processes that correspond to actions in the structure diagrams. The function step inserts the functions into the model as executable operations that produce required outputs. Finally, the purpose of system timing is to specify constraints on the implementation.

- **Software Quality Deployment:** Software Quality Deployment (SQD) is a quality system for client satisfaction. It investigates needs and requirements of the client with the aim to improve the software system and the development process. SQD is driven by the voice of the customer. The method uses a series of matrices to deploy customer and quality requirements. These deployments are a structured way of dealing with a special concern at a detailed level, and provide a basis for linking these concerns into an integrated framework. SQD identifies, explicitly states, and prioritizes requirements. The SQD process builds a series of matrices correlating user needs and technical requirements. The matrices consist of the client characteristics/client segments matrix, the client segments/client requirements matrix, the client requirements/technical requirements matrix, and technical requirements/entities-and processes matrix. Firstly, customers and users are identified and classified in client segments. The problem is outlined and the development horizon is defined. Then primary needs of the customer are documented and refined. Information about needs can come from various sources, e.g. customer opinion surveys and market research data. The requirements are then used in client segments to understand which segments value which requirements and to what extent. This matrix indicates whether defined requirements cover client needs and depicts conflicts between needs and requirements. It also provides an elaborate method to prioritize requirements on the basis of their importance to the customer.
- **Soft Systems Methodology:** Soft Systems Methodology (SSM) views Requirements Engineering as a recurrent cycle that aims at improving the problem situation. It applies systems thinking to explore the problem situation and to specify the software systems. SSM assumes that different views of the application domain exist and that these views must be considered while generating requirements. The method compares perceptions of the problem with models of its application domain. Initiated by this comparison, assumptions about the problem are drawn, challenged and tested. SSM is based on the assumption that RE proceeds via debate and that it is intrinsically participative. Initially, the requirements engineer enters the situation considered problematic and identifies the clients for the software system to be built. A first investigation of the problem situation is carried out and documented. Next, root definitions are formulated. These root definitions specify the purpose of the system in terms of transformation processes. In constructing root definitions, one has to consider the mnemonic CATWOE, where the C stands for customer as the victim or beneficiary of T the transformation process. A denotes the actors who perform the activities of the system, whereas O is the owner who can stop these activities, and E are the environmental constraints. W describes the worldview when formulating root definitions. The next stage of SSM compares conceptual models with perceived reality using informal discussion, formal questioning, scenario writing based on operating the model, and trying to model the problem in the same structure as the conceptual model. Finally, the comparison stage uses the differences between models and reality to discuss changes, which are desirable and feasible concerning the worldviews specified in root definitions. Implementing these changes completes the SSM cycle.

Methodology Comparison

Table 1:

Requirement Engineering Methodologies

Sr. No.	*Methodology*	*Characteristic*	*Requirement Engineering Process Coverage*
1	**Core-Controlled Requirements Expression**	Viewpoint Oriented Analysis	Requirement Analysis Requirement Elicitation
2	**Issue-based Information System**	Tracking Issues and proposition thereof	Requirement Elicitation Requirement Analysis
3	**Jackson System Development**	Formalizes Functional Specification development in 5-step process from initial requirements	Requirement Specification Requirement Analysis
4	**Software Quality Development**	Using series of matrices, phases and deliverables of the Requirement Engineering process and mapped and traced	Requirement Elicitation Requirement Analysis Requirement V & V
5	**Soft Systems Methodology**	Represents requirements in the form root definition, built as per the acronym CATWOE	Requirement Elicitation Requirement Analysis Requirement Specification

***5.* Models and Frameworks:** Models and Frameworks provide a skeleton and overarching structure that guides the process progression. This section presents models and frameworks that have been inspirational to the development of the Synergistic Requirements Model.

- **RGM:** The RGM proposes a refinement to the conventional requirements generation process, by providing a structured framework that first distinguishes between requirements generation and requirements analysis, and then recognizes distinctively different activities within requirements generation, i.e. instruction, preparation, elicitation and evaluation. The latter three activities are embodied within an iterative process focusing on requirements capturing. The RGM provides guidelines and enforces protocols to help structure its attendant activities. Guidelines are suggestions or recommendations that offer support by serving in an advisory capacity. Protocols are rules that establish boundaries through pre-defined constraints. The RGM provides a means of structuring the requirements capturing process by providing a distinctive framework that guides the activities through which the customer and requirements engineer define requirements. It also incorporates a monitoring methodology that ensures all requirements elicitation activities follow proper procedures.
- **Requirements Triage:** Knowing what not to build is as important as knowing what to build. Davis claims that the software product planning helps extract the right product features, which is imperative for a development effort to be successful. In this spirit, the Requirements Triage advocates the consideration of factors such as price, market, cost, time-to-market, market size, feature mix, market penetration, revenues, profits and return on investment as a key to project success. After an initial selection of capabilities to be included in the product, the selection needs to be verified accounting

for development, sales, marketing and finance perspectives. If any of these perspectives indicate a losing situation, then it can be fathomed that the company will lose, resulting in a possible failure of the project. The process commences by eliciting features. Based on an elaborate set of guidelines, the features are categorized and prioritized based on the relative importance, cost and risk. Next, a decision is made about the features that are worth including in the final specifications, and the ones excluded are deferred to later releases. This is done by simply drawing a line among the sorted and prioritized list of features, so that the sum of the efforts required for the features above the line is less than or equal to the tentative development budget. If an acceptable situation is encountered, then software product planning is accomplished. On the other hand in case something does not fit in, either one of the criterion may be adjusted to find a satisfactory balance.

- **Knowledge-Level Process Model:** According to the composition, the elicitation process provides initial problem descriptions, requirements and scenarios elicited from stakeholders, as well as domain knowledge. The manipulation of requirements and scenarios process resolves ambiguous, inconsistent requirements and scenarios not supported by needs. This process reformulate requirements from informal requirements and scenarios, to more structured semi-formal requirements and scenarios. It also provides relationships amongst requirements and scenarios. The maintenance process of requirements and scenarios specification is used to store and retrieve the documents in which the information requirements and scenarios are described, including information on traceability.
- **The Win-Win Spiral Model:** Following are the main constituent processes of the Win-Win Spiral Model determine objectives: Identify process constituents and establish system boundaries and external interfaces determine constraints, determine the conditions under which the system would produce a win-lose or lose-lose outcome identify and evaluate alternatives, solicit suggestions from sources, evaluate them with respect to constituent's win conditions, analyze, assess and resolve win-lose or lose-lose risks record commitments, identify and record problematic areas that may require attention in the future stages of the development life cycle, cycle through the Spiral, resolve risks, prototype, accumulate responses and perform another iteration through the spiral, if needed.

Model Comparison

Table 2:

Requirement Engineering Models and Frameworks

Sr. No.	*Model*	*Characteristic*	*Requirement Engineering Process Coverage*
1	**Requirements Generation Model (RGM)**	Structure the process of capturing requirements from the customer	Requirements Elicitation Requirements Verification
2	**Requirements Triage**	Extensive Requirements Analysis, based on right product features	Requirements Analysis Requirements V &V
3	**Knowledge-level Process Model**	Separations of concerns aid process abstraction	Requirements Elicitation Requirements Management
4	Win-Win Spiral Model	Continuous, Collaborative development, refining requirements with each cycle	Requirements Management Requirements Analysis

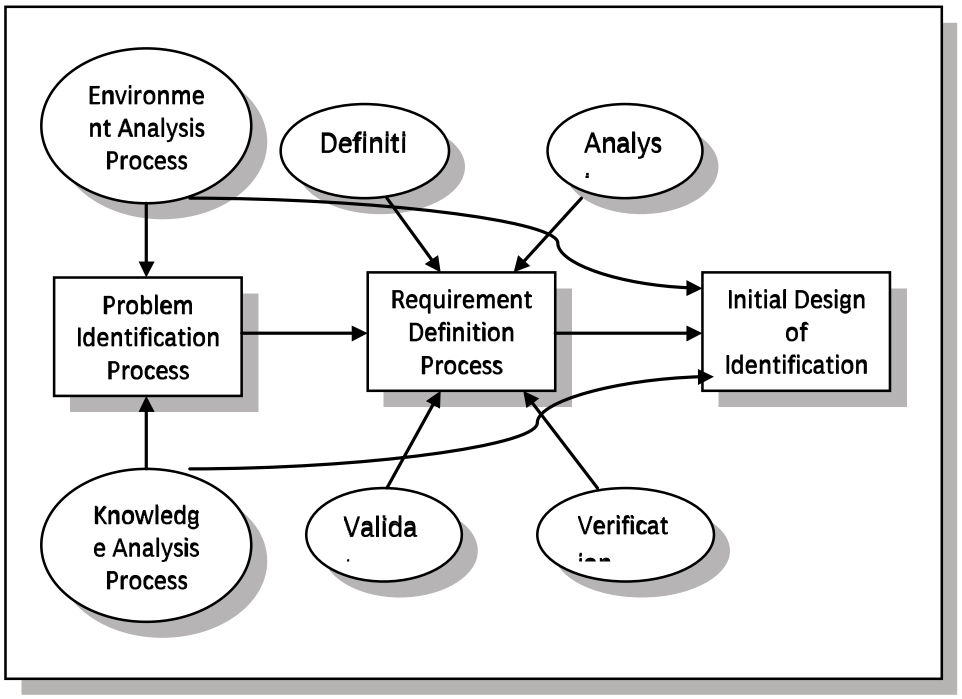
6. Requirement definition Process

The goal of the requirement definition process is to transform the stakeholder requirement into a set of technical requirements.

- **Solution Definition Process:** The Solution Definition Process is used to generate an acceptable design solution for Logical Solution Requirement, the developer shall define one or more validated sets of logical solution representation that conform with the technical requirements of the system
- **System Analysis Process:** In the analysis process, the developer shall perform risk analysis to develop risk management strategies, support management of risks and support decision making. The step of risk analysis can generate some safety requirement other than that defined by the acquirer and stakeholder. These new requirement must be taken into account.
- **Requirement Validation Process:** Requirement Validation is critical to successful system product development and implementation. Requirements are validated when it is certain that they describe the input requirements and objectives such that the resulting system products can satisfy them.
- **System Verification Process:** The System Verification Process is used to ascertain that the generated system design solution is consistent with its source requirements, in particular, safety requirements.

Figure:3

Process of Initial identification model design


Environme nt Analysis Process
Definiti
Analys
Problem Identification Process
Requirement Definition Process
Initial Design of Identification
Knowledg e Analysis Process
Valida
Verificat

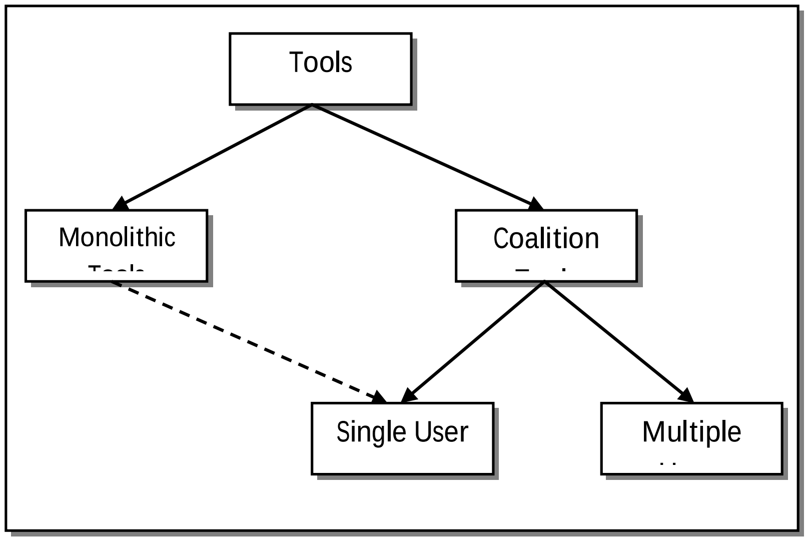
7. Software Development Tools: Tools that support software development can be classified into two groups: the first group contains monolithic development benches created and supported by a single vendor or organization. These development benches target a single platform, and are designed to support as much of the whole software development process as possible. Monolithic tools generally only support specific development technologies and target platforms. They are large, very complex and do not provide flexibility. They are costly to build and acquire, hard to maintain and modify for different goals. Monolithic tools are developed by a single group, and are inherently integrated. They aim to support the whole software development process, but generally support a fraction of it in practice.

The second group is composed of individual tools supporting one or more discrete software development phases. Software development organizations targeting different technologies or those operating in heterogeneous environments require a variety of tools. Monolithic tools do not provide support for a mix of target technologies and are not suitable for modification.

Tools are used to handle the complexity surrounding software development processes. There is much evidence in the literature on how tool use provides benefits for software development in terms of quality and cost.

Figure:4

Types of tools



Conclusion

One of the principal problems of traditional software development lies in the fact that those who have been primarily involved in software development to date have not been willing to recognize that software development is, in most cases, mainly a question of occupational and/or organizational planning. Whenever software development is to be approached from such a perspective, it would be planned from the beginning to engage experts in occupational and organizational planning in the process of software design.

References / Bibliography

1. Alford ,M. W.(1977). *A Requirements engineering methodology for real-time processing requirements*, IEEE Transactions on Software Engineering, **3**(1):60-69 January 1977.

2. Arthur ,James D.(2003). *A Synergistic Approach to Software Requirements Generation: The Synergistic Requirements Generation Model (SRGM), and An Interactive Tool for Modeling SRGM (itSRGM)*, Virginia Polytechnic Institute and State University; May, 2003.

3. Babb, R. and L. Tripp(1980). *Toward Tangible Realizations of Software System* ,Proc. of 13th Hawaii Intl. Conf. on System Sciences, January 1980.

4. Beizer B.(1983). *Software Testing Techniques*, New York :Van Nostrand Reinhold.

5. Bell, T., et al.(1977). *An Extendable Approach to Computer-Aided Software Requirements Engineering,* Trans. Software Eng., vol. SE-3, (1) January 1977.

6. Chaczko, Zenon, Quang Jenny and Moulton ,Bruce(2010). *Knowledge Transfer Model for the development of Software Requirements Analysis CASE Tolls to be Used in Cross Time- Zone,* University of Technology, Sydney, Australia, Feb 2010

7. Conradi, R. and Westfechtel B.(1998). *Version Models for software configuration Management*, ACM Computing surveys 30 (2): 232-282, June 1998.

8. Conte ,S. D., Dunsmore H. E. and Shen ,Y. E.(1986). *Software Engineering Metrics and Models*, Benjamin/ Cummings, Menlo Park, CA.

9. Davis , A. M. (1995). *Software Prototyping.* In Advances in Computer, vol. 40, pp: 39-63. Academic Press,

10. Dubosis E., Hagelstein, J. and Rifant ,A.(1991).*The Requirements Engineering of Computer System*, ed. Thayse, A. John Wiley & Sons Ltd.

11. Erturkmen ,K. Alpay(2010). *A process modeling based method for identification and implementation of software development tool integration-Tuple*, The Middle East Technical University, March 2010

12. Fairley ,Richard(1997). *Software Engineering Concepts*, New Delhi:Tata McGraw-hill Publishing Company Limited.

13. Finkelstein ,A. (ed)(2000). *The Future of Software Engineering*, New York: ACM Press.

14. Ghezzi ,Carlo, Jazayeri Mehdi and Mandrioli, Dino(2002). *Foundation of Software Engineering*, New Delhi :Pearson Education (Singapore) Pvt. Ltd. , 2nd Edition.

15. Heinonen, S.(2006). *Requirements Management Tool Support for Software Engineering in Collaboration*. University of Oulu, Department of Electrical and Information Engineering. Master's Thesis.

16. Heninger, K.(1980). *Specifying Software Requirements for Complex Systems: New Techniques and their Application*, IEEE Trans, Software Eng., vol. SE-6(1) January 1980.

17. IEEE(1987). *Software Engineering Standards*. Technical Report.

18. Jackson ,D. and Rinard ,M(2000). *Software Analysis: A Road Map*, available at http://people.csail.mit.edu/rinard/paper/icse00.pdf

19. Jalote ,Pankaj(2011). *An Integrated Approach to Software Engineering*, New Delhi: Narosa Publication House, 3rd Edition.

20. Lloyd ,Wes J.(2001). *Tools and Techniques for Effective Distributed Requirements Engineering: An Empirical Study*, Virginia Polytechnic Institute and State University, July 11, 2001

21. Pressman ,Roger S.(1997). *Software Engineering: A Practitioner's Approach*, McGraw Hill International Edition, 6th Edition

22. Schwalbe, Michael(2011).*The 40-30-30 Rule: Why Risk is Worth It*, The 99 Percent. Web. 2 January 2011. Available at http://the99percent.com/tips/6103/The-40-30-30-Rule-Why-Risk-Is-Worth-It

23. Stephen T. Freeza (1995).*Requirements Based Design Evaluation Methodology*, University of Pittsburgh

24. Summi Yasuyuki, Hori Koichi and Ohsuga Setsue(1998). *Supporting the acquisition and modeling of requirements in software design, Knowledge-Based Systems*

25. Wiegers, Karl(2003). *Software Requirements. Redmond: Microsoft Press*, available at http://www.processimpact.com/pubs.shtml

26. Wolak ,Ronald G.(2010). *DISS 725 – System Development: Research Paper 4 Software Process Assessment and Improvement Models*, Graduate School of Computer and Information Sciences, Nova Southeastern University, July 2010

27. Yong, R.(2001). *Effective Requirements Practices*, Addison-Wesley .

28. Zultner. R.(1992). *Quality function development for Software Satisfying customers*, American Programmer, February 1992, pp 28-41.