Volume 3, No. 5, May 2012
Journal of Global Research in Computer Science
ISSN-2229-371X
TECHNICAL NOTE
Available Online at www.jgrcs.info

INITIAL CONCEPT HOW TO DESIGN A SOFTWARE REQUIREMENT IDENTIFICATION MODEL

Sudhir Singh[1], Dr. Raj Kumar[2]
[1]Research Scholar, Mewar University, Gangrar, Chittorgarh, Rajasthan
sudhirb26@gmail.com
[2]Director, Dr Rizvi College of Engineering, Kaushambi, Allahabad

Abstract: Standard does not purport to address all safety problems associated with its use or all applicable regulatory requirements. It is the responsibility of the user of this standard to establish appropriate safety and health practices and determine the applicability of regulatory limitation before use. We design pre and post conditions that help identify and bound the problem and then present some methods and technology that assist in refining that boundary and also in recognizing essential characteristics of the problem. Problem identification focuses on going an understanding of the customer's problem domain and identification 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 to shift to the description of what computers are required to solve as opposed to how computer are to solve a particular problem. In this paper we propose a system for aiding the construction of a software requirement model through the techniques, that is a method for the support of idea formation and knowledge based modeling.

INTRODUCTION

At the same time, because of the need to apply computer system to a wide variety of domains, it has become essential to integrate target domain experts, i.e. the system users along with computer experts, into the process of determining specification of requirements. However, it is very difficult to ascertain a system's complete requirements from users who are not expert in software development, let along transform these into formal specifications in the software design.
A requirement space is a metric space which enables a user to visualize requirement concepts and their relationship to the user's original ideas, mainly consisting of abstract and domain dependent expressions. A requirement model is an initial software model, which aids in the formulation of a user's requirements to succeeding phases such as details specification and implementation.

THE HISTORICAL PERSPECTIVE

The original Waterfall Model introduced by Royce outlines a sequence of activities that are still found in most of today's software development process. Leading the development process is the Requirement Analysis activity, followed by Design, Coding and Integration & Testing. The conceptual framework provided significant insights as to how software should be developed. Furthermore, it paved the easy for the definition of many similar models and paradigms that are composed of essentially the same basic set of activities e.g. The Object Oriented and Spiral Models. Requirement engineering can be divided into 2 mains groups of activities. (1) Requirement Development: this activity includes the processes of elicitation documentation, analysis and validation of requirements. (2) Requirements managements: this activity includes process of maintainability management, changes management and requirements traceability.

PROBLEM IDENTIFICATION PROCESS

The objective of the Problem Identification is to gain agreement on the problem definition. Common obstacles that stand in the path of meeting this objective are

(a) The customer has only a cursory understanding of the problem
(b) The customer is convinced of a problem formulation that is inconsistent with the symptoms.
(c) The customer is thinking in the solution space before he/she gained any understanding of the underlying problem and finally
(d) The requirement engineer's lack of domain knowledge

Once the analyst and customer have agreed on the problem in principle, the analyst needs to produce a formal Problem Statement. Leffingwell and Widrig provide an outline of the format of such a statement. More specifically, the problem statement must

(a) Provide a description of the problem elements
(b) Identify stakeholders affected by the problem
(c) Describe the impact of the problem on the stakeholders and business activities, and
(d) Indicate the proposed solution along with few key benefits.

The two conceptual models describe how the solvers, users and software knowledge engineers vies the environment the relations between relevant concepts and software product requirements. It is important for all of the viewpoints to converge in a single representation of the system. This representation shows what the software is to do and when and how it is to do it and what knowledge it is to use

(a) Environment Analysis Process: This process is performed to set the software in its external environment. It is especially valid when the software is to be embedded in a bigger system. The statement of need is the basis for environment

analysis, identifying the inputs, required outputs and full system functions.

(b) Knowledge Analysis Process: The knowledge analysis process seeks to define all the existing concepts, attributes and functions, which generates a static and dynamic knowledge structure, enabling the engineer to represent his/her understanding of solver knowledge and the solver to identify conceptual errors on the part of the engineer; this structure is referred to as a knowledge model.

REQUIREMENT DEFINITION PROCESS

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

(a) 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

(b) 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.

(c) 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.

(d) 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.

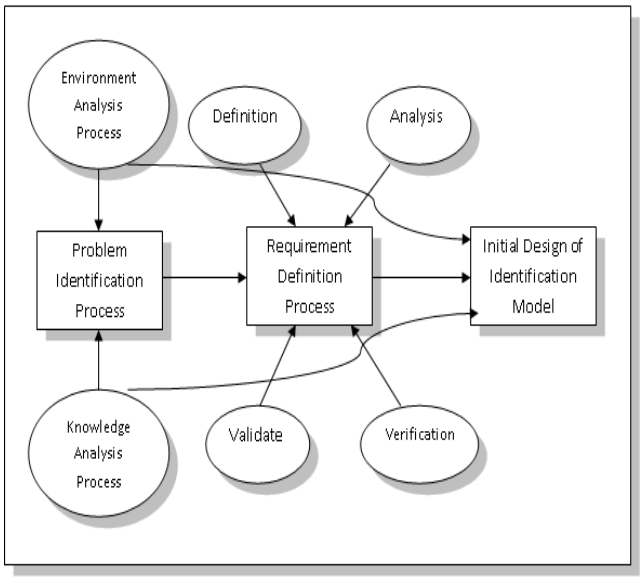

Figure:1 Process of Initial identification model design

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. Where software development 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] IEEE, software engineering standards. Technical report, 1987.

[2] A. M. Davis. Software Prototyping. In Advances in computer., val. 40, pages 39-63. Academic Press, 1995.

[3] E. Dubosis, J. Hagelstein and A. Rifant " The Requirements engineering of computer System", edited by A. Thayse, John Wiley & Sons Ltd, 1991.

[4] Stephen T. Freeza "requirements Based Design Evaluation Methodology, University of Pittsburgh, 1995.

[5] Zultner. R., "Quality function development for Software Satisfying customers", American Programmer, February 1992, pp 28-41.

[6] Yong, R. Effective Requirements Practices, Addison-Wesley 2001.

[7] Babb, R. and L. Tripp "Toward Tangible Realizations of Software System" Proc. 13th Hawaii Intl. Conf. on System Sciences, January 1980.

[8] Bell, T., et al.: "An Extendable Approach to Computer-Aided software Requirements engineering," Trans. Software Eng., val. SE-3, no. 1 January 1977.

[9] Heninger, K.: "Specifying Software Requirements for Complex Systems: New Techniques and their Application," IEEE Trans, Software Eng., val SE-6, no. 1 january 1980.

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

[11] B. Beizer, Software Testing Techniques, Van Nostrand Reinhold, New York 1983.

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

[13] Yasuyuki Summi, Koichi Hori, Setsue Ohsuga, " Supporting the acquisition and modeling of requirements in software design, Knowledge-Based Systems 11(1998)

[14] S. D. Conte, H. E. dunsmore and Y. E. Shen, Software Engineering Metrics and Models, Benjamin/Cummings, Menlo Park, CA, 1986.

[15] A. Finkelstein (ed), The Future of Software Engineering, ACM Press, New York, 2000.

[16] D. Jackson and M. Rinard, "Software Analysis: a RoadMap" in Finkelstein, 2000.

[17] Wiegers, Karl. Software Requirements. Redmond: Microsoft Press, 2003. Print. online at: http://www.processimpact.com/pubs.shtml

[18] Zenon Chaczko, Jenny Quang, Bruce Moulton, "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

[19] Ronald G. Wolak, "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

[20] Schwalbe, Michael. "The 40-30-30 Rule: Why Risk is Worth It." The 99 Percent. Web. 2 January 2011. Online at: http://the99percent.com/tips/6103/The-40-30-30-Rule-Why-Risk-Is-Worth-It

[21] Pankaj Jalote, "An Integrated Approach to Software Engineering", Narosa Publication House, New Delhi, Third Edition.

[22] Carlo Ghezzi, Mehdi Jazayeri, Dino Mandrioli, "Foundation of Software Engineering", Pearson Education (Singapore) Pte Ltd, New Delhi, Second Edition.

[23] Roger S. Pressman, "Software Engineering", McGraw Hill International Edition, Sixth Edition.

[24] Richard Fairley, "Software Engineering Concepts", Tata McGraw-hill Publishing Company Limited, New Delhi.

