Problem Identification and Decomposition within the Requirements Generation Process

Ahmed S. Sidky, Rajat R. Sud, Shishir Bhatia and James D. Arthur
Department of Computer Science, Virginia Tech
Blacksburg, VA 24060, USA

{asidky, rsud, shbhatia, arthur}@vt.edu

ABSTRACT

Only recently has the real importance of the requirements generation process and its requisite activities been recognized. That importance is underscored by the evolving partitions and refinements of the once all-encompassing (and somewhat miss-named) Requirements Analysis phase of the software development lifecycle. Continuing along that evolutionary line, we propose an additional refinement to the requirements generation model that focuses on problem identification and its decomposition into an associated set of user needs that drive the requirements generation process. Problem identification stresses the importance of recognizing and identifying the difference between a perceived state of the system and the desired one. We mention pre- and post-conditions that help identify and bound the "problem," and then present some methods and techniques that assist in refining that boundary and also in recognizing essential characteristics of the problem. We continue by presenting a process by which the identified problem and its characteristics are decomposed and translated into a set of user needs that provide the basis for the solution description, i.e, the set of requirements. Finally, to place problem identification and decomposition in perspective, we present them within the framework of the Requirements Generation Model.

Keywords: Requirements Engineering, Problem Analysis, Problem Identification, Problem Decomposition

1. INTRODUCTION

A system is only as good as the requirements from which it is developed. It is crucial, therefore, to gain a firm understanding of the customer's stated as well as implied requirements. While significant strides have been made in developing tools and techniques, in acquiring such an understanding, the formal concepts, frameworks and processes underpinning the use of those tools and techniques are often ill-defined, or simply lacking. For example, we tout the need for Requirements Validation – but what do we validate the requirements against? Problem Analysis is a second example. More specifically, we would ask: *What constitutes Problem analysis? Where does it fall within the requirements generation process? What are the activities supporting it?* In this paper we focus on answering the above questions because we are convinced that effective Problem Analysis is crucial to the evolution of requirements that meet both the stated as well as the intended needs of the customer.

Gause and Weinberg [6] characterize a "problem" as "the difference between things as perceived and things as desired." *Problem Analysis*, therefore, can be viewed as the process of understanding the customer's real problem, and then translating that understanding into a set of needs. Reflecting this view, we have identified two principal activities within Problem Analysis – Problem Identification and Problem Decomposition. *Problem Identification* focuses on gaining an understanding of the customer's problem domain and identifying the root cause(s) of the symptoms being observed by the customer. That root cause(s) *is* the problem. *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, i.e., requirements.

The remainder of this paper elaborates on what we have outlined above. In Section 2 we provide a more comprehensive discussion motivating the concepts behind and the need for Problem Analysis. In Section 3 we provide an in-depth examination of the issues and activities associated with both Problem Identification and Problem Decomposition. Section 4 provides a description of specific tools and methods that assist the analyst in Problem Analysis activities. We conclude by outlining where Problem Analysis fits within the *Requirements Generation Model* developed at Virginia Tech [1].

2. BACKGROUND AND MOTIVATION

The importance of a well-defined, effective set of requirements engineering activities is constantly being reinforced as we begin to recognize the manifold relationships between the quality of a product and the quality of requirements from which it is developed. The success rates for our development efforts, however, have been abysmal. According to the Standish Group, five of the top eight reasons why projects fail are related to the requirement generation process – incomplete and/or erroneous requirements, lack of user involvement, unrealistic customer expectations, and requirements volatility [12]. *If requirements are so important then, why are we only now beginning to define (or refine) processes and activities that support their synthesis?* An examination of how requirements generation has evolved over time, and how it has been repressed helps us answer this vital question.

The Historical Perspective

The original Waterfall Model introduced by Royce [11] outlines a sequence of activities that are still found in most of today's software development processes. Leading the development process is the Requirements Analysis activity, followed by Design, Coding, and Integration & Testing. This conceptual "framework" provided significant insights as to how software should be developed. Furthermore, it paved the way for the definition of many similar models and paradigms that are composed of essentially the same basic set of activities (or phases), e.g. The Object Oriented [3] and Spiral Models [2]. A closer examination of existing software development models reveals a common thread: they all contain a Requirements Analysis phase, it is usually one of the first phases in the model, and it is composed of a series of activities that relate to the gathering and analysis of requirements from the customer. Although "first" in the sequence of phases, Requirements Analysis has been last in line for re-examination and refinement. In fact, the redefinition and refinement of the original phases defined by Royce proceeded in a backward fashion starting with Integration & Testing. We attribute this to the fact that as we move from Requirements Analysis toward the Integration & Testing phase, conceptualizing and addressing indigenous issues become relatively easier. Moreover, when the software engineering community did begin examining and refining the Requirement Analysis phase, its somewhat inappropriate name contributed to the perception that activities therein, like Problem Analysis and Requirements Elicitation, were only minor ones [4]. Subsequently, only within the last few years have we seen a meaningful refinement of "Requirements Analysis" that recognizes the major activities that underlie requirements generation. Because it is the opinions of the authors that *Requirements Generation* is a better characterization of those activities performed in the requirements phase, we will use it in lieu of Requirements Analysis.

Refining the Requirements Generation Process

The Software Engineering Community now recognizes five major activities with the Requirements Generation process: Requirements Elicitation, Requirements Verification & Validation, Requirements Specification, Requirements Analysis and Requirements Management [13, 14]. Within this set we once again find the phrase "Requirements Analysis." In this new context, however, it is defined to be "the process of analyzing the customer and user needs to arrive at a definition of software requirements" [13]. And once again we believe this phase is being ill-defined because, by definition, it could also include verification and validation activities – clearly this was not intended. When considered relative to the other four major requirements generation activities, however, we can infer that Requirements Analysis is composed of those activities that deal with *problem analysis*, feasibility assessment, and risk analysis and assessment.

Figure 1 places Problem Analysis in perspective. That is, it follows Concept Definition but precedes Requirements Elicitation. The Concept Definition activity is not considered a requirements generation activity. Its primary objective is to produce a document that provides a conceptual overview of how the system might operate. This document is called the Concept of Operations document, and is used as input to Problem Analysis. Problem Analysis can be considered as the first principle activity of the requirements generation phase.

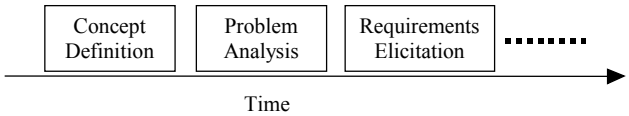

Figure 1

Distinguishing Between the Problem and Solution Spaces

Before going further, it is important to recognize one additional distinction implied by Figure 1 – the difference between formulating a description of the customer's problem and formulating a description of a solution to that problem. As stated earlier, a problem is the difference between things as perceived versus things as desired. According to Gause, the most common starting point for requirements generation is to start thinking in terms of a solution before identifying the problem that the solution is going to solve [7]. Such thinking, however, can lead to solutions to the wrong problem, or to a solution that is unnecessarily constraining. Problem Identification, the first activity within Problem Analysis, focuses on confirming that the problem being conveyed is, in fact, the root cause of the observed (and often undesirable) symptoms. Once identified (and confirmed), the problem is then recorded as a single descriptive unit that includes all necessary clarification and qualification information, i.e., a problem statement. A set of customer needs is then determined by decomposing the problem statement into a set of smaller, refined problem components. The problem statement and set of derived needs define the problem space. That is, they define what the *customer* needs.

The set of customer needs are used as input to the requirements elicitation activity. It is here that each need is translated into one or more (software) system requirement, which when implemented, will solve the customer's problem. In effect, the set of elicited requirements define the solution space – that is, they define what the system *shall* do.

3. PROBLEM IDENTIFICATION AND DECOMPOSITION

In Section 2 we have provided a historical view of Requirements Generation and have presented an evolutionary path to the Problem Analysis activity. We have also identified (a) where Problem Analysis fits into the Requirements Generation process, and (b) the two principal activities comprising it, i.e., Problem Identification and Problem Decomposition. The objective of Problem Analysis, and subsequently a common one for Problem Identification and Problem Decomposition is to develop a comprehensive understanding of the problem perceived by the customer, and to identify the corresponding needs that, in turn, will serve as the basis for requirements elicitation. We now examine the individual components of Problem Analysis.

Problem Identification

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 has gained any understanding of the underlying problem, and finally
(d) the requirements engineer's (or analyst's) lack of domain knowledge.

Educating the requirements engineer in the existing and proposed system as well as the operational environment helps mitigate the adverse impact of (d). The other three obstacles are customer oriented and require an investigation into the perceived inadequacies of the current system (if one exists at all) and the corresponding operational needs.

As alluded to earlier and as implied in Figure 1, a cursory description to the problem is provided by the Concept of Operations document. Alternatively, depending on the magnitude of the software development effort, a statement of the problem can take the form of a set of high-level requirements coming from a prior *Systems* Engineering effort. It is our experience that the effort associated with Problem Identification is greater in case of the former as compared to the latter. This stems from the fact that the set of high-level requirements are obtained as a result of a formal analysis activity during Systems Engineering, while the structure and content of the Concept of Operations document may vary widely. In either case, the correctness of the incoming document and what it implies must be examined before proceeding. In other words, we must determine if the problem it outlines is, in fact, the root cause of the observed symptoms. We use the following example to help clarify this concept.

A manager receives numerous complaints from users about the poor performance of his organization's web service. The manager contacts the web administrator and states that he needs to place the web service on a faster machine. Being an experienced analyst, he/she recognizes that the manager has just provided a solution without stating the problems. So, rather than starting a search for a more powerful web server, the analyst asks "*What problem will a more powerful web server solve?*." The manager then explains that users are complaining about the poor response time for the web services. After applying root cause analysis, the real problem is identified as a recently installed search algorithm which takes $O(n^2)$ time rather that the expected O(n log(n)). Replacing the inefficient algorithm with a more appropriate one led to a more acceptable response time.

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 [9]. 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 a few key benefits.

On a final note, although tools and techniques to support Problem Identification do exist, they often reflect a multi-purpose flavor. In the next section we discuss several supporting techniques, including Root Cause Analysis and the use of Context Free Questions.

Problem Decomposition

Once the actual problem is identified the next step in Problem Analysis is to decompose the problem into smaller distinct elements, and to further refine their individual characteristics. The intent is to gain additional insights into the problem, and subsequently, a better understanding of the customer needs. Decomposition and refinement are activities of Problem Decomposition.

Problem Decomposition involves a series of steps by means of which *a set of needs* is obtained, from which the requirements are derived. Like its predecessor, Problem Decomposition is an iterative process that begins with root cause analysis. This presumes, of course, that the actual problem has been identified and has a well-defined problem statement. In addition, we recommend that the analyst be educated about the problem-specific aspects of the customer's domain and the environment within which the new/modified system will eventually function. This indoctrination either outlines or provides the basis for (a) defining the solution boundaries and (b) identifying the constraints to be imposed on the solution space, e.g., economic, political, technical, etcetera.

As mentioned above, Problem Decomposition is an iterative process that often involves numerous meetings with the customer. Prior to each meeting the analyst needs to ensure that he/she has identified the appropriate set of stakeholders (or meeting participants), set a meeting agenda, and has outlined each participant's role and responsibilities prior to the meetings. To support problem refinement, each meeting must have a focused objective and employ structured activities that support the achievement of that objective, e.g., recording decomposition components and evolved needs, and monitoring/controlling the meeting process. Finally, at the end of each meeting, the identified "set of needs" are evaluated relative to their correctness, completeness and non-ambiguity. The final set of customer needs can be "validated" (in a loose sense) against the Con-Ops document or against the set of high-level requirements formed during the systems engineering process. As illustrated in Figure 1, the set of customer needs are then provided as input to the requirements elicitation process and form the basis from which the requirements (or solution specification) are derived.

4. TOOLS AND METHODS SUPPORTING PROBLEM ANALYSIS

We have identified various tools and methods that support Problem Identification and Problem Decomposition. In general, most of those we have encountered can be applied to both Problem Identification and Problem Decomposition; very few exist that deal with them as separate entities.

Nonetheless, how do we identify tools that support either of the two activities, or perhaps both? The answer is based on the observation that Problem Identification relies on *searching* for one or more unknowns, and then when found, applying a *refinement* activity to the discovery. Clearly, these activities require a substantial amount of customer interaction. Problem Decomposition, on the other hand, is predominately an *analysis* activity focused on extracting customer needs. Hence, the set of tools and/or methods that emphasize customer interaction can be applied to Problem Identification; the set dealing with analyzing, decomposing and refining a problem is more suited to Problem Decomposition. We do note that customer

interaction is also present during Problem Decomposition, but just not as much as is required for Problem Identification.

During our investigation of Problem Analysis we have identified several applicable tools and methods. A majority of them are interaction-oriented and support various interaction formats with the customer. A representative set of tools and methods that can be applied to *both* Problem Identification and Decomposition are:

- Interviews,
- Preliminary System Study,
- Cause Effect Diagram and
- Root Cause Analysis.

A representative set applicable primarily to problem identification are:

- Context Free Question and
- Walk Through Technique.

A principal tool supporting the objectives of problem decomposition is

- Problem Frames.

Figure 2 shown below provides an illustration of the process underlying Problems analysis and attempts to place these tools and methods relative to that process.

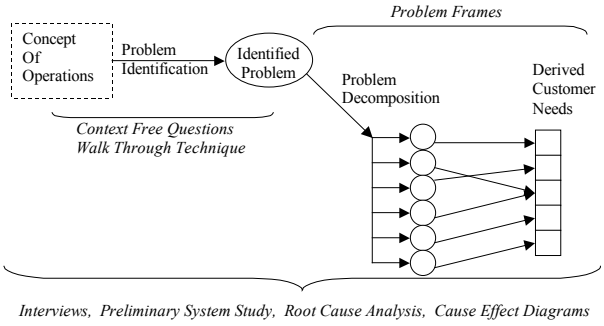

Figure 2

*Context Free Qu*estions and *Interviews* can serve as effective tools for problem identification. Context free questions can be viewed as "high-level questions that can be posed early in a project to obtain information about global properties of the design problem" [7]. Context free questions like "*Who is behind the request for the system?*", "*Is there another source of solution that you need*?", "*What problems does this system solve?*" are especially effective during initial meetings and interviews because they set the stage for problem identification. Later in the interview process, context free questions like "*Are my questions relevant to the problem your have?*", "*Are you the right person to ask the questions?*", and "*Are your answers official?*" add to the validity of this approach to problem identification. The facts and figures revealed during the use of context free questions often provide significant insights into the real problem.

Cause-effect analysis and diagrams assist in understanding the causal relationships that exist in a system and which, in turn, help during problem decomposition. Cause-effect diagrams are a reflection of the 'divide and conquer' technique. Using a cause-effect diagram, the identified problem (or symptom) is broken down into its constituent parts contributing to the problem. In effect, we are simply decomposing the problem into smaller system elements, each of which is viewed as a potential candidate causing the perceived symptom or problem. Figure 3 illustrates how the Cause-Effect (or Fishbone Diagram) is used to decompose the perceived problem into potential causes of that problem. Using an example mentioned earlier in the paper, we observe four potential causes for slow response to web page requests, i.e., a slow CPU, network overload, slow disk access and/or an inefficient search algorithm. The "effect" is the slow response to web page requests; the (potential) causes are the four components mentioned above. In the final analysis, the decomposed problem also serves as the basis from which we begin identifying the set of customer needs.

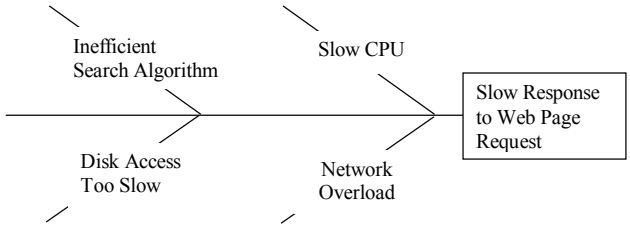

Figure 3
The Fishbone Diagram

Root Cause Analysis, Preliminary System Study and Cause-Effect Analysis are often used in conjunction when identifying the actual problem, and decomposing it into its principal components. A flowchart outlining the relationship of these methods within the framework of problem identification and decomposition is provided in Figure 4.

The *Walk Through* (WALT) technique described by Lenart [10] is another method that assists in problem analysis. WALT uses the Concept of Operations Document provided by the customer as its input. Within WALT the customer is guided (or "walked") through a description of the perceived problem. As the description unfolds, the analyst records the description and constituent components. The analyst continuously asks questions in increasing detail in an effort to gain an understanding of (a) the problem as a whole, and (b) the potential causes of the problem.

Finally, the concept of *Problem Frames* is examined. Problem Frames are used in the Jackson System Development process and employs problem decomposition as a method to identify potential reuse components [8]. Having identified the problem, Problem Frames focuses on decomposing the problem into solution pieces that are known to already exist. Hence, this tool can be used to extract from the original problem those components that have the existing solution templates. This is an example of a "solution-oriented" approach to problem decomposition.

The tools and methods discussed above help support problem identification and/or problem decomposition. Many other tools exist, of course, that assist in these two aspects of Problem Analysis. There is, however, one piece of this puzzle that particularly needs additional exploration and tool/method development – generating customer needs from elements of the decomposed problem set. Currently this activity relies mostly on intuition and experience.

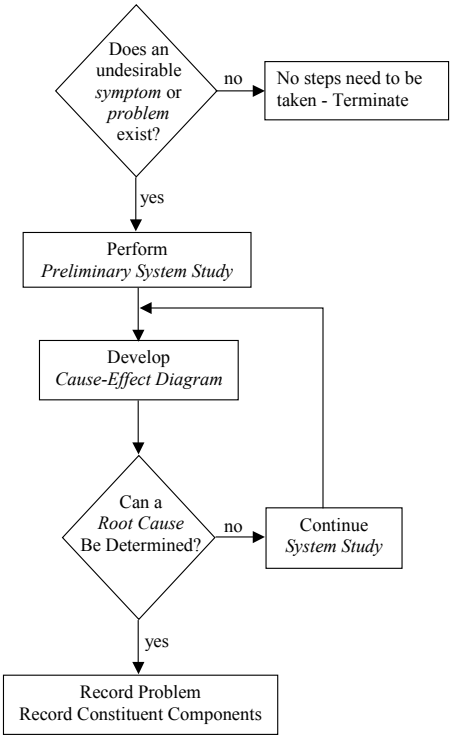

Figure 4
Relating System Study, Cause-Effect Diagrams and Root Cause Analysis

5. PLACING PROBLEM ANALYSIS WITHIN THE REQUIREMENTS GENERATION MODEL

The primary task of the requirements engineer or analyst is to capture accurately the requirements expressed by the customer. One of the many impediments to achieving this goal lies within the customer's domain. That is, the customer often (a) has only a minimal understanding of the many functions that the proposed system is to support, (b) lacks experience in identifying and expressing desirables as requirements, and (c) states desirables in terms that lead to misleading or incorrect inferences by the requirements engineer [5]. In similar ways, the requirements engineer also contributes to the challenge, e.g., lack of domain knowledge. Taken together, they all contribute to the potentially difficulty of the task that must *precede* requirements capturing, i.e., *Problem Analysis*.

*So, where does Problem Analysis **really** fit into the requirements generation process?* Figure 1 provides a very high-level picture that attempts to answer that question. The Requirements Generation Model (RGM) shown in Figure 5, however, provides a more detailed illustration that places Problem Analysis in relation to the other major activities that support requirements generation. Because a substantial number of RGM activities rely on the artifacts produced during Problem Analysis, and because we desire to place Problem Analysis within an holistic framework, we provide a brief overview the RGM next.

The RGM refines the requirements analysis phase of the conventional Waterfall Software Development Model by imposing a framework and methodology that structures the problem analysis, requirements elicitation, recording and evaluation processes [1]. In particular, it partitions the requirements analysis phase into an initial problem analysis phase followed by a minimally overlapping indoctrination phase. (The overlap is present because educating the requirements engineer [or analyst] about the perceived problem is part of the indoctrination phase, yet it is a necessary precursor to completing Problem Analysis.) The indoctrination phase is then followed by an iterative requirements capturing phase.

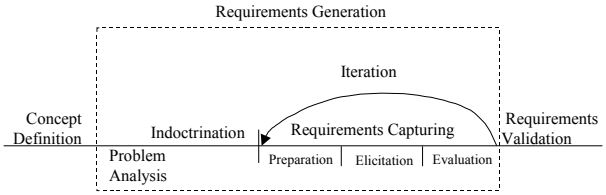

Figure 5
The Requirements Generation Framework

As stated earlier, the objectives of the problem analysis phase are (a) to evolve a succinct and accurate description of the problem and (b) to derive a set of customer needs. The objectives of the indoctrination phase are to (a) introduce the customer to the requirements definition process, (b) provide the requirements engineer with an overview of the customer's problem domain and needs, and (c) describe the participants' tasks and responsibilities in the requirements definition process. The objectives of the requirements capturing phase are directly related to the three sub-phases that comprise it – preparation, elicitation and evaluation. Respectively, the primary objectives of those sub-phases are to (a) define the scope of the elicitation meeting and ensure that all participants have completed their pre-meeting assignments, (b) enable the requirements engineer to accurately identify and record software requirements as expressed by the customer, and (c) evaluate the contributions of the preceding elicitation meetings, identify unresolved (or new) issues, and determine if an additional refinement iteration is needed. The RGM also includes a monitoring methodology that operates throughout the requirements generation phase and in tandem with attendant activities to ensure that proper procedures and protocols are being followed.

Clearly, a successful Problem Analysis phase is crucial to the success of the remaining RGM phases. More specifically, the succeeding phases cannot hope to meet their objectives unless a well-formulated set of customer needs can be derived. That set of needs is predicated on identifying the root cause of the problem, and then successfully decomposing it into its contributing constituents.

6. SUMMARY AND CONCLUSIONS

The "Chaos" study has shown the detrimental impact of developing systems that have ill-defined requirements. A substantial contributor to ill-defined requirements is attempting to generate requirements before the causal problem has been identified, studied and understood. In this paper we have provided an overview of the Problem Analysis process. Through its two major activities, Problems Identification and Problem Decomposition, that requisite understanding of the problem can be captured. We have also discussed several of the tools and methods supporting Problem Analysis, and have provided examples outlining how they can work together in a synergistic fashion, as well as how they can be used independently. Finally, using the Requirements Generation Model as a foundation we are able to place the Problem

Analysis process in relation to other activities supporting requirements generation, and in doing so, illustrate why Problem Analysis is so crucial to evolving requirements that reflect those intended by the customer.

7. References

[1] Arthur, J.D. and Markus K. Groener (1999). "An Operational Model Supporting the Generation of Requirements that Capture Customer Intent," Proceedings of the Pacific Northwest Software Quality Conference, Portland OR, October 1999, pp. 286-302

[2] Boehm, B.W. (1988). "A Spiral Model of Software Development and Enhancement," *IEEE Computer*, Vol. 21, No. 5, May 1988, pp. 61-72.

[3] Booch, G. (1986). "Object-Oriented Development," *IEEE Transactions on Software Engineering*, Vol. 12, No. 2, February 1986, pp. 221 ff.

[4] Davis, A.M., *Software Requirements: Objects, Functions, & States*, Prentice-Hall, Upper Saddle River, New Jersey, 1993.

[5] Freedman, DP and G. M. Weinberg (1990). *Handbook of walkthroughs, inspections, and technical reviews: evaluating programs, projects, and products*, 3rd ed. New York, NY: Dorset House Publishing Co., Inc., 1990.

[6] Gause, D.C. and G.M. Weinberg (1989a). *Are Your Lights On? How to Know What the Problem Really Is*, Dorset House Publishing, New York, 1989.

[7] Gause, D.C. and G.M. Weinberg (1989b). *Exploring Requirements: Quality Before Design*, Dorset House Publishing, New York, 1989.

[8] Jackson, D. and M. Jackson (1996). "Problem Decomposition for Reuse," *The Software Engineering Journal*, Vol. 11, No. 1, January 1996, pp. 19-30.

[9] Leffingwell, D. and D. Widrig (19XX), *Managing Software Requirements: A unified Approach*, Addison-Wesley Object Technology Series, Addison-Wesley, Boston MA, 2000.

[10] Lenart, M. and Ana Pasztor (1998). "A Participatory Design Requirement Engineering System," www.cs.fiu.edu/~pasztora/design/padre/aid98.ps

[11] Royce, W.W. (1970). "Managing the Development of Large Software Systems: Concepts and Techniques," *Proceedings of the Western Electronic Show and Convention (WesCon)*, Los Angeles, August 1970, pp. 1-9 (Reprinted in the *Proceedings of 9th International Conference on Software Engineering*, March 1987, Monterey CA, pp. 328 – 338.)

[12] Standish Group (1995), "Chaos", Standish Research Paper, http://www.standishgroup.com/sample_research/chaos_1994_1.php

[13] Thayer, R.H. and M. Dorfamn (1997). *Software Requirements Engineering*, 2nd Edition, IEEE Computer Society, Los Alamitos CA, 1997.

[14] Young, R.R. (2001), *Effective Requirements Practices*, Addison-Wesley Information Technology Series, Addison-Wesley, Boston Mass, 2001.