USER NEEDS AND REQUIREMENTS, AND LIFE SUPPORT SYSTEM SPECIFICATIONS

Palmer, James D.
Information Technology and Engineering, George Mason University, USA

Keywords: Requirements, user needs, elicitation, assessment, risk analysis, prototype, traceability, specifications, transformation, requirements process

Contents

Summary

User needs and requirements provide essential information concerning the development and construction of any system to assure that the developed system meets the stated needs of the originating organization. The acquisition and articulation of this information is the most critical part of a systems development process and may be especially difficult due to the very complexity of the process of acquiring and analyzing user needs and requirements. Collection and analysis of information must be correct, consistent and unambiguous. Added to this complexity is the question of being certain that the information may be validated and accessible to those from the community of interest. A process to develop user needs and requirements to meet these attributes is fundamental. User needs and requirements for life support systems, initiated by user determination that a need exists that is not met by existing systems, require such a

disciplined approach. System level requirements are developed to broadly outline the desired capabilities, which, in turn, are investigated to ascertain feasibility and practicality and examine trade-offs. Once the feasibility and practicality of the desired system have been determined to be necessary and sufficient to launch a new system (or significant modification of an existing or legacy system), the resulting specifications are analyzed for errors and any errors found are resolved, difficult or incomplete concepts are prototyped, and the final product is transformed to formal or semi-formal languages of a specification. The resulting product is an acceptable set of user needs and requirements.

1. Introduction

The first step in the development of any system is to determine just what the user wants, the purpose of the ensuing system, and the necessary accompanying requirements. User needs and requirements represent the first and most important step in systems development and must be clear, concise, consistent, unambiguous, and represent user goals and purposes for the system to be constructed. An organized approach to produce user needs and requirements is the preferable method and will surely result in a more concise and complete explanation of what is to be constructed than an ad hoc scheme. Thus, a process for user needs and requirements articulation is the initial step in systems development. Any process used to acquire and articulate user needs and requirements must ensure the aforementioned qualities, as best as possible, in the resulting specification documentation. User needs and requirements describe the system to be constructed as seen by the user and include agreement across and between users and developers as to what it is that is to be constructed and provides, as well, a basis for estimation of cost and time to complete. The products of a user needs and requirements process include definition of the product to be constructed, documented specifications of this product, any constraints or environmental concerns that must be met, performance needs, interface requirements, and quality attributes of the system. Minimally, such a process to acquire and articulate user needs and requirements must include: 1) elicitation of the purpose(s), constraints, functions, and intended utilization environment; 2) assessment of the validity, risk and constructability of the consequent system; and 3) transformation of these statements to the concise language of a specification satisfactory for design and implementation.

The development of life support systems depends on accurate and correct requirements to assure the system developed is the one needed and works in the manner required by the user. This is true for the entire range of possible systems from simple to complex. These systems form the bases of natural or human engineered systems that provide long term sources of water, energy, and food and include systems such as those used to extract sediment samples from the ocean floor that enable geologists to ascertain the health of the seas to integrated automated fingerprint identification systems for human security to monitoring the environment for pollutants to enable environmental scientists identify the health of the environment in which we live to health support systems for human care to construction of human habitats. Each of these complex systems has at least one common development attribute; i.e., a set of requirements and specifications that were prepared that enabled construction of the intended system. This common attribute includes information on the performance needed to carry-out the mission, the

functional (or behavioral) aspects of the system, and the non-functional (or non-behavioral) features. Non-functional requirements include management activities, quality factors, operating environment, and other non- behavioral aspects. Successful system development depends on the ability to satisfy these needs and to reflect them in the delivered system. User needs and requirements that are complete, correct, consistent, and error free, play a major role in ensuring that the delivered system meets the intended purpose. Critical keys to understanding the relationships that exist within and across system requirements, design, construction, test, and implementation are to be found in these user needs and requirements specifications. Correct and accurate requirements are so critical to the delivery of a system that meets the needs of users, is on time and within budget, that the process by which these requirements is developed is a critical element in systems design and implementation activities.

Requirements generation is perhaps the most critical part of the development process. And, requirements are especially difficult to specify due to the very complexity of the systems that are intended for construction. For example, in the instance of an environmental monitoring system, diverse aspects must be considered; such as, obtaining permission to place monitoring equipment on private land, assuring that data collection is consistent and repeatable, performing the collection of meta data around the sites and be able to accomplish all of this in real-time. Added to this complexity is the question of being certain that the data is consistent so that it may be placed in a common database and permit access by those from the community of interest.

Life support systems are initiated by user determination that a need exists that is not currently being met by existing systems. From this beginning, system level requirements are developed to broadly outline the desired capabilities, which, in turn, are investigated to ascertain feasibility and practicality and examine trade-offs. Once the feasibility and practicality of the desired system have been determined to be necessary and sufficient to launch a new system (or significant modification of an existing or legacy system), the resulting specifications are analyzed for errors and any errors found are resolved, difficult or incomplete concepts are prototyped, and the final product is transformed to formal or semi-formal languages of specification. It is essential to have correct and complete requirements to assure that the delivered system meets the stated organizational needs of the user. The notion that desired capabilities lead to requirements is a basis for current interests in capability based planning.

While there is widespread acceptance of the necessity to provided adequate requirements development, there is often considerable controversy as to the ultimate need, purpose, and cost of performing requirements activities necessary to assure that the delivered product meets user needs. The controversy arises primarily because of knowledge acquisition and technical difficulties, the lack of automated approaches to implement the processes, and the concomitant time and effort that must be utilized to apply any of the presently available support tools. Developers simply do not see the benefits that may accrue to the final product, when the entire requirements process is fully implemented, compared to the time and effort required. Users generally do not understand the complexity of transforming their concepts to requirements documents that developers can use. The problems and issues that surround the requirements development process are discussed in the next section.

Production of well-understood and well-developed requirements information, and the identification and management of risk in the requirements engineering phase, is a paramount concern to assure production of systems that meet user needs with a system delivered on-time and within budget. It has been indicated by a number of researchers that somewhere between 50% and 80% of the errors found in delivered systems can be traced to errors in interpretation of requirements developed during the requirements process. Thus, this process becomes the pacing activity in the development of large complex engineered systems.

2. Problems and Issues Concerning Requirements Development

Difficulties related to requirements development generally revolve around the necessity to elicit or develop requirements information from a variety of sources; e.g. users and/or existing systems. There is usually a lack of definitive information from the user and a lack of agreement on mechanisms to transform these concepts to documents for use by the developer. These difficulties lie at the interface between the system developer and the user in the identification of system level requirements. Once the requirements documentation is received from the user, this must be transformed to the exact language of specification; another source of potential problems. There are also potential technical difficulties that relate to hardware, performance, capacity, interfaces, and other issues.

Requirements development processes are often misunderstood by users, frequently misapplied by developers, and seldom performed correctly by either group. Issues and concerns emanate from the complexity of a project itself that must be confronted when implementing requirements development. Each discipline; such as, environmental monitoring systems, automated fingerprint systems, sediment extraction systems, or health monitoring systems will undoubtedly have languages, methods, and tools peculiar to the discipline. This results in a potential lack of ability to utilize the same language constructs across the interrelated viewpoints necessary for the development of these systems, which, in turn, may lead to errors in the development of requirements used to provide linkages within and across these viewpoints. Some of the issues that need to be addressed by the user and developer at the time of system development include: how to apportion projects by discipline, the type and nature of information that should be available and used across different disciplines, and the types of tools that can be used to provide consistent and correct requirements across disciplines. Establishing threads across disciplines is simply difficult due to language, method, and tool differences.

Generally, system level requirements are stated in natural language which includes a high potential for incorporation of ambiguities, conflicts, inconsistencies, and lack of completeness, simply due to differences in natural language usagees. These problems must be addressed prior to transformation from the informal language of users to the formal or semi-formal languages of design. Otherwise, these deficiencies may be incorporated in the final product rather than be resolved to reflect the specific needs of the user at the time of requirements development. If this occurs, these deficiencies must be corrected during later stages of system development or following system delivery, provided the system is considered to be acceptable at all by the user. Either of these approaches results in considerably higher costs in terms of person-hours and funds than

does detecting and correcting errors during the requirements stage.

Since the products of requirements have little or no direct consequence to the development team, other than indicate to them what is to be constructed, the development of proper requirements elements often has had a low priority with developers. However, the benefits of correct requirements are seen directly at the onset of the development life cycle, they are necessary for designs to be developed, become essential during validation testing and system installation and operation, and then are used directly for integration tests to assure users that the correct system has been developed. Thus, the utility and need for a user needs and requirements development process that will instill confidence that the requirements are correct and as error free a possible is readily apparent.

3. Requirements Process

A user needs and requirements development process is needed for verification and validation that the user needs are properly interpreted and to assure that the products that result are those intended by the user. This process enables access to information concerning the system to be developed and provides visualization and understanding into the techniques necessary for system development. Requirements are needed to validate the impact of changes, provide process control, and enable early risk management. It permits insights to components such as quality, consistency, completeness, impact analysis, system evolution, and process improvement. It is equally important to have the capability to trace a requirement to its origin, as well as test the results of the construction to assure compliance with the intent of the user, as well as for identification and resolution of risk, development of appropriate integration tests, and the delivery of a project that meets the needs statements of the requirements.

Proper requirements development addresses problems related to assessment of under or over designs; investigation of high level behavior impact; review of non-functional requirements; and conflict detection. With correct requirements development, there is the assurance that decisions made later in the system development life cycle are consistent with earlier decisions. Test cases that check coverage for module and integration testing and for requirements validation are provided by appropriate requirements. Proper requirements provisions provide the basis for the development of an audit trail for the entire project by enabling the establishment of links within and across system entities, functions, behavior, performance, etc.

3.1 Problems and Issues to be addressed by the Requirements Development Process

Of the many concerns to be addressed by the requirements engineering process, those that are listed below are the most common, difficult to resolve, necessary to settle to assure the delivered system is the one intended by the users.

1. conflict within and across requirements;
2. lack of consistency of requirements individually and in clusters of similar statements;

3. incompleteness across requirements and their clusters;
4. inability to determine the ripple effect impact of adding new requirements to existing systems;
5. issues related to storage and retrieval;
6. volatility in requirements generation;
7. failure to provide for traceability matrices for complete auditing of all requirements throughout life of the project (including maintenance); and
8. requirements that may be technically and economically unfeasible.

The key to a successful requirements engineering process is the resolution of the problems and issues that are so much a part of the requirements process at the time of requirements development, not after deficiencies are noted during design or testing. An approach to resolve these is to utilize a system development life cycle for the requirements process that includes techniques to address the problems and issues.

3.2 Definition of Terms Used in a Requirements Process

There are many terms that are used to describe or delineate requirements or that relate to requirements. Some of these terms correlate to the how and why of requirements, while others connect to the outcomes or what of requirements. The usual meaning of the term is given first, while the last meaning given is in the context of systems requirements and an example of usage.

1. Allocation: The act of distributing; allotment or apportionment; as to assign or apportion functions to specific modules.
2. Behavior: The way in which a system acts, especially in response to a stimulus; stimulus- response mechanisms; as, activity or change in functions across sub-systems.
3. Classification: A group of entities ranked together as possessing common characteristics or quality; the act of grouping or segregating into classes which have systematic relationships; a systematic grouping of entities based upon some definite scheme; as to classify requirements according to organizational or performance characteristics.
4. Cluster: things collected together, a bunch; a group; a grouping of entities based on similarity; as to cluster requirements according to language syntactical information.
5. Function: The characteristic action or the normal or special action of a system; one aspect of a system is so related to another that there is a correspondence from one to the other when an action is taken; as an algorithm to provide the equations of motion.
6. Hierarchy: A series of objects or items divided or classified in ranks or orders; as in a type of structure in which each element or block has a level number (1= highest), and each element is associated with one or more elements at the next higher level and lower levels; as a single high level requirement decomposes to lower level requirements and to design and code.
7. Meta Data: Data about data; as to provide data such as time of day, location coordinates, and meteorological considerations at an environmental data collection site.

8. Requirement: A requisite condition; a required quality; to demand; to claim as by right or authority; to exact; as to demand system performance by the user.
9. Thread: To connect as to pass a thread through; string together, as to link behaviors of a system together.
10. Traceability: The course or path followed; to follow or track down; to follow or study out in detail or step by step, especially by going backward over evidence (Ed: as to trace design from requirements); to discover or uncover by investigation; as to trace to the source; as to follow requirements from the top level to design and implementation and back.
11. Tree: A diagrammatic representation which indicates branching from an original stem; as system components derived from a higher level entity to more discrete lower level entities.

3.3 A System Life Cycle Process for Requirements Management

There are many life cycle models and one of the simplest is the interactive and iterative life cycle. In this life cycle the project is initiated with the development of the concept which states what the overall system should do. As complete an understanding is developed as possible at this time, including system objectives and any constraints and variables that are present. Following this, a definition of the overall functional form of the system is delineated. Once the system concept has been determined and agreed to by the key participants, the detailed designs and functional specifications are prepared. The next steps are to develop system architecture to accommodate the functional specifications, design the system, build and test the system, conduct integration testing and evaluation, and finally, implement and maintain the system. For any system development life cycle, the system requirements specifications must be fully explicated to assure that the system to be constructed is understood by both user and developer. These life cycle activities require documentation of needs and outcomes. This life cycle will be used as the model for the requirements engineering process life cycle.

The system concept, system objectives, constraints, variables, and functional and non-functional specifications are generally prepared in an informal language (English in the instance of the United States). Usually, the language utilized for user needs and requirements development is natural language. Use of natural language introduces problems such as conflicts, ambiguities, inconsistencies, and incomplete statements. Thus, it is most difficult to utilize informal languages and not introduce vagueness and conflicts that result simply from the characteristics of the language itself.

This development of the system concept is perhaps the most difficult task for life support systems. It is an activity that involves not only hardware and software, but also human interaction, definition of constraints, statement of variables, and a significant range of other parameters that must be considered. To accomplish the development of a system concept, it is desirable to follow a relatively rigorous process, such as that depicted in Figure 1. This plan minimally must include the development of system objectives, refinement of these objectives, development and refinement of system constraints and variables; expressing as concisely as possible a high level functional model; formulating a design and implementation strategy; and documenting the outcomes of these activities in the form of design specifications. The requirements

engineering process model includes all of the essential functions that are necessary for the development of requirements. These activities include elicitation; organization of materials in a logical order, analyses of the user developed statements for consistency, errors, omissions, risk, and completeness; modeling difficult to understand concepts; and transforming the informal language of users to the formal or semi-formal languages of design. A process model for requirements engineering is shown in Figure 1.

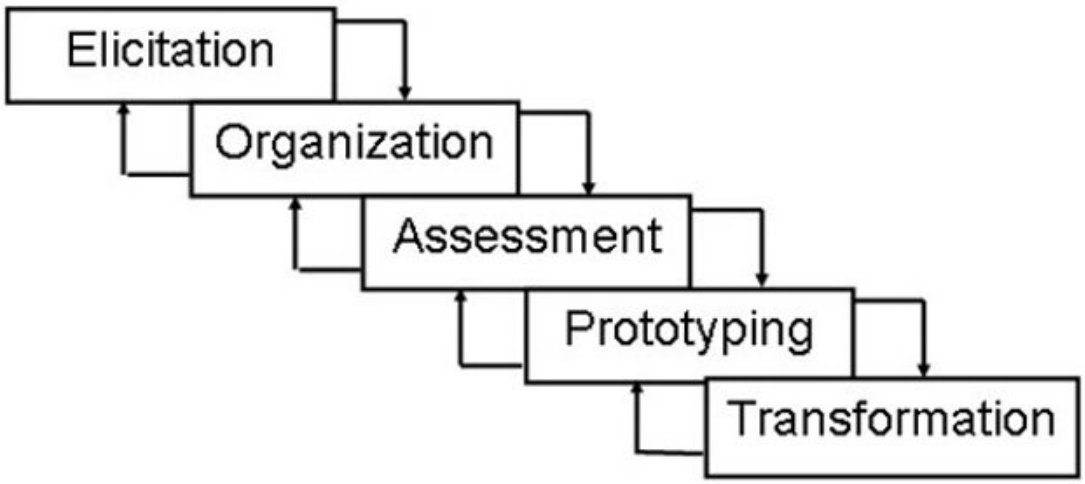

Figure 1: Requirements Engineering Process Life Cycle

The steps that are to carried out include elicitation of the needs, constraints, variables, and objectives from the user and the overall management of requirements. Following this development of the system concept, the requirements statements are organized so as to cluster activities together that belong together or are related in some fashion, based on characteristics that are selected by the user and developer. These could include similarities related to function, objective, or algorithm and the like. Next a study of the statements is conducted to determine the presence of errors, issues, or missing aspects of the system. Prototypes, as appropriate, may be constructed to provide high level functional models of the system. Finally, the statements are transformed to the concise language of specifications.

-
-
-

Bibliography

Andriole, Stephen J. (1996). *Managing Systems Requirements: Methods, Tools and Cases*. New York:

McGraw-Hill. [A comprehensive coverage of methods, approaches and tools for systems management].

Boehm, B.W. (1981). *Software Engineering Economics.* Englewood Cliffs, New Jersey, USA: Prentice-Hall. [A classic treatise on approaches to measurement of the costs of software production].

Davis, A.M., (1993). *Software Requirements: Objects, Functions, and States.* Englewood Cliffs, New Jersey: Prentice Hall. [A fine book on software requirements processes and procedures].

Committee on Naval Analytical Capabilities and Improving Capabilities-Based Planning, *Naval Analytical Capabilities: Improving Capabilities-Based Planning*, National Academies Press, Washington DC, 2005 [One of several recent studies of the development and use of capabilities for requirements and planning.]

Palmer, James D., and N. Ann Fields (1992). An Integrated Environment for Software Requirements Engineering. *IEEE Software*, May 1992, 80-85. [This covers environments and methods for the development of user needs and requirements].

Palmer, James D. and YiQing Liang (1992). Indexing and Clustering of Software Requirements Specifications, *Information and Decision Technologies*, 18, 283-299. [An application of classification methods to user needs and requirements].

Sage, Andrew P., and James D. Palmer (1990). *Software Systems Engineering*. New York: John Wiley & Sons, Inc.[An excellent source of material on systems approaches to software engineering].

Sage, Andrew P. and William B. Rouse (1999). *Handbook of Systems Engineering and Management. New York*: John Wiley & Sons. [Contemporary coverage of nearly all topics of interest for the development of user needs and requirements and systems engineering].

Sage, Andrew P. (2005). "Systems Architecting," *McGraw Hill Yearbook of Science and Technology*, 2005, pp. 358-361. [An overview ofnsystems architecture, including the Zachman and other architectural framework.]

Salton, G. and McGill, M.J. (1983). *Automatic Information Organization and Retrieval.* New York: McGraw- Hill Publishing. [A seminal work on classification systems].

Sommerville, Ian and Pete Sawyer (1997). *Requirements Engineering: A Good Practices Guide*. Chichester (UK): John Wiley & Sons. [A review of good contemporary practices for requirements engineering].

Thayer, Richard H. and Merlin Dorfman, editors (1997). *Software Requirements Engineering.* Los Alamitos, CA: IEEE Computer Society Press. [This presents a comprehensive coverage of all aspects of user needs and requirements engineering for software and is generally applicable to general systems as well].

Biographical Sketch

James D. Palmer, Professor Emeritus, Information Technology and Engineering, George Mason University, received the BSEE in 1955 and an MSEE in 1957 from the University of California, Berkeley, CA and the PhD in Electrical Engineering in 1963 from the University of Oklahoma, Norman, OK. He has served as Professor of Electrical Engineering and Director of the School of Electrical Engineering of the University of Oklahoma, Professor of Electrical Engineering and Dean of Science and Engineering of Union College, Schenectady, NY and President of Metropolitan State College of Denver, Denver, CO. He was the founding Administrator (Assistant Secretary), Research and Special Programs Administration, the United States Department of Transportation. He served as Vice President of Research and Development, Mechanical Technology, Inc., Latham, NY and Executive Vice President, JJ Henry Co, New York and Moorestown, NJ. He was the first holder of the BDM International Chair in Information Technology at George Mason University where he held the positions of Associate Dean for Research and Graduate Studies, Acting Dean of the School of Information Technology and Engineering, and Chair, Systems Engineering Department. He has been co-author of three books and more than 100 papers in the areas of systems engineering, software systems engineering, and user needs and requirements. He received the degree Doctor of Public Service, *honoris causa*, from Regis University, Denver, CO. He is listed in many bibliographical reference works including Who's Who in America, American Men and Women of Science, and Who's Who in the West. He has received many honors including Outstanding

Service Awards for contributions as President from the Systems Man, and Cybernetics (SMC) Society of the IEEE, the Joseph G. Wohl Outstanding Career Award from SMC, the Millennium Medal for Outstanding Achievements and Contributions from the IEEE, and the Outstanding Public Service Medal and Award from the United States Coast Guard for service as member and Chair of the United States Coast Guard Academy Advisory Committee (Board of Visitors).