SYSTEM REQUIREMENTS

Buys R.T.
CMS Information Services, Inc., USA

Keywords: user requirements, system specifications, requirements elicitation, prototypes, rapid application development (RAD), joint application development (JAD), facilitated application specification techniques (FAST), integrated product teams (IPT), interviews, questionnaires, use case, expert systems

Contents

Summary

The definition of system requirements begins with the user, who can represent many points of view and impose numerous constraints on the system. Many of the techniques described include features that help resolve discrepancies in points of view. In many domains it is critical that the systems engineer understands the technical and operational characteristics in addition to the functions carried out by the users. This is important for the requirements identification and specification process because unachievable needs should not be included in the specification. The specific objective of the requirements

phase is to create a work product that can be used by the design team and test team to develop and test the system.

Interviews are a staple for requirements elicitation. They are almost always part of the process even if not the major requirements gathering technique selected by the systems engineers. Interviews are an effective requirements identification technique that has been employed by systems analysts on many different types of program. Questionnaires can be very effective when there are a large number of users and if users are geographically disbursed. Facilitated application specification techniques (FAST) can include marketing, finance, and sales as well as the functional and technical experts. Joint application development (JAD) techniques bring teams together to develop requirements and initial specifications. An objective is to get the people most affected by the product involved from the beginning so they can contribute to the entire development cycle.

Prototypes are approximations or models of the final product. Prototypes can be created prior to meeting with users, as is the case with rapid application development (RAD) methodologies, or they can be built after the first set of requirements and specifications are gathered, as a way to be sure the requirements are correct. They are particularly effective in communicating the understanding of the users' needs to the users and/or customers and to the design team. Integrated product teams (IPT) include representatives from functional disciplines who can make decisions and get products defined and implemented. Artificial intelligence and expert systems can bring new methods to bear in uncovering user requirements and system specifications that are critical to a successful system implementation. Supporting tools and techniques can enhance the ability of a method to uncover user requirements and system specifications by clarifying difficult interfaces or highlighting unusual processing requirements.

1. Introduction

Producing systems, particularly software-intensive systems that meet user and performance requirements, have been acknowledged as a significant challenge in modern system development. Understanding user requirements is important because of the impact errors in this phase have on the overall cost of the delivered system. If developers misunderstand user requirements, it can add thousands of dollars to the cost of the system to correct it when the user discovers the error after delivery. The increased attention on the requirements phase has led to a plethora of methods and techniques for identifying requirements and describing them in sufficiently specific terms that they can be modeled, designed, and implemented in subsequent phases of the life cycle. This article begins with a discussion of the issues involved in determining user requirements. Then it presents an overview of different approaches for identifying user requirements and system specifications, and concludes with a short description of several documentation techniques for a systems specification.

2. Identifying System Requirements

The definition of system requirements begins with the user who can represent many points of view and impose numerous constraints on the system. The job of the systems

analyst is to make sure all points of view are included and constraints considered in defining requirements. There are also categories of requirement that must be identified. Users have a set of functions or capabilities that they want, but they also want certain levels of performance from the system. In addition, the analyst must consider how requirements can be validated, whether functions are to be defined in phases, and how to merge the different input into a coherent, testable statement of system requirements.

The following sections address these issues and provide a framework for assessing the completeness of an approach for identifying and specifying system requirements.

2.1. Points of View

One of the biggest challenges facing analysts in defining system requirements is the need to include all the users and stakeholders in the requirements elicitation process. Each of these participants brings a unique point of view to the process that incorporates different needs, priorities, constraints, environmental factors, funding options, and political realities. The task of the systems analyst is to merge these different points of view into a coherent statement of requirements. The process of merging different points of view will reveal conflicts among requirements, overlaps, redundancies, and omissions among the requirements. These differences must be resolved before the requirements specification is complete.

Many of the techniques described in Section 4 include features that help resolve discrepancies in points of view. The FAST and IPT techniques and prototypes, for example, help ensure that all stakeholders are included so that there are no omissions in the requirements. Group discussion formats can help resolve conflicts and redundancies as well. Understanding that different points of view exist and being aware of the need to seek them out and merge them into the systems requirements specification is critical to the success of the specification. Analysts who do well at defining system requirements typically have very good interpersonal skills as well as strong technical and analytical skills. Stakeholders bring different backgrounds and skills to the process that requires that the analyst be able to interact on many levels of technical sophistication. If the analyst is aware of the different points of view, it will help him or her to recognize conflicts and facilitate their resolution to the satisfaction of all.

2.2. Functional, Nonfunctional, and Other Requirements

Systems requirements are of different types. Often analysts will group them into at least two categories —those describing the functions of the system and those describing how well the system is to perform those functions. When describing the functions of the system, the task is to describe the external behavior of the system (therefore, these are sometimes called behavioral requirements) or what the user expects the system to do as opposed to how the system is to do it (i.e., the behavior inside the system). It is often not possible to completely eliminate consideration of how the system is to perform its functions from the requirements specification, but that should be the goal of the systems analyst. The objective is to leave as much discretion to the designer—who will get the specifications next—as possible so the best design can be created for the system. By the same token, neither analysts nor designers should add functions to the system that are

not approved by the customer or stakeholder. Adding unneeded functionality increases cost that the customer has no obligation to reimburse.

Describing how well the system is to perform its functions brings a host of quality considerations into the specification. Quality factors that may be considered are usability, reliability, response time, portability, maintainability, defects, availability, human computer interface (HCI), ergonomics, and many others.

2.3. Understanding the Domain

In many domains it is critical that the systems engineer understand the technical and operational characteristics in addition to the functions carried out by the users. An in-depth understanding is usually not required, but a basic appreciation of the priorities, constraints, and environmental factors that influence the way the users perform their functions can be critical to the development of a good system requirements specification, and can save time and money in the development process.

When systems analysts understand the domain they will be better able to identify requirements that will have the most value to the product. For example, in identifying requirements for a control system for vessels, analysts would find it useful to know that barge traffic along the Mississippi River often skims the mud bottom without adverse repercussions. In this domain, it may not make sense to include a requirement for an automatic braking mechanism when the hull comes within a certain measurement of the river bottom. If the domain includes harbors with rock bottoms, this might be desirable, and a user with a different point of view might propose it. The role of the systems analyst is to be sufficiently aware of the domain to sort through the needs expressed by different users and narrow the list down to the appropriate set.

One way to acquire understanding of the domain is to engage a sufficient number of users in the requirements identification process. When different points of view are represented, the opportunity exists to uncover unique aspects of the domain that may have a major influence on the requirements. A number of the methodologies and techniques described in Section 4 involve the inclusion of different users groups and facilitate the resolution of the different points of view they bring to the discussion of system requirements.

2.4. Feasibility of a Solution

In many domains, the ability to pursue certain solutions will be limited by a variety of factors. These include budgets, environmental constraints, political considerations, technology, laws, and standards. This is important for the requirements identification and specification process because unachievable needs should not be included in the specification. Going from the Earth to the Moon in two hours is not technically feasible at this time. Discussion with the user may reveal other more achievable objectives that could be expressed in a requirements specification. However, going from the Earth to the Moon in two hours would require years of basic research that may or may not yield a feasible solution to the requirement. Even if the user is willing to finance such research, it is clear that the task is not one of engineering an achievable solution for

near-term use.

When moving from an understanding of the problem to a description of the user's functional requirements to a specification, the feasibility of a solution to the problem becomes more and more relevant. In large systems engineering projects, these steps may be identified as distinct phases without outputs that are defined and documented separately. In other projects, these steps are part of a single phase, with the requirements specification as the output, reflecting constraints and enabling one or more feasible solution(s).

2.5. Constraints

In almost all cases there are limits on what a system is allowed to do or how it is to be built. Similar to the factors that may limit the feasibility of a solution (see Section 2.3), budgets, environmental factors, political considerations, technology, laws, and standards may represent one or more constraints on the implementation of a system.

Constraints are often uncovered through requirements elicitation activities. Users and other participants bring their concerns as well as their needs and desires to requirements gathering sessions. The concerns may reflect constraints and should be analyzed for possible inclusion in the specification. Constraints are included in the requirements document for several reasons. First, they should be validated with affected groups. Including them in a requirements specification ensures that they will be part of formal or informal review sessions with the user and other engineering groups. Since constraints impact the use or implementation of a system, designers need to be able to review the constraints as they review the requirements for the system. In addition, incorporating constraints into the requirements specification is an opportunity to identify and resolve any conflicts or inconsistencies with the functional or performance requirements.

2.6. Validating and Verifying Requirements

Validating requirements ensures that they represent the needs and desires of the user and can be shown to exist in the work products and final system. According to Bahill, the objective is to build the right system and show that it does what it is supposed to do. Validation determines the correctness of an end product, compliance of the system with the customer's needs, and completeness of the system. Bahill goes on to say that validating requirements ensures that the set of requirements is consistent, that a real-world solution can be built that satisfies the requirements, and that it can be proven that such a system satisfies its requirements.

Verifying requirements proves that a requirement has been satisfied. This is usually done through inspection, test, simulation, or other means. If prototypes and pre-production units have been built, the requirements must be verified for each of these interim stages, as well as in the production units.

The terms are often defined to support individual system engineering environments. Several sources indicate that it is necessary to be sure the specific interpretation and

definition of the terms is understood by everyone on the team to avoid confusion. Being sure the product is the right one and that it is created the right way are both important indicators of quality, and should be confirmed no matter what term is used to describe them.

3. Requirements Identification Strategy

In the past project managers expected to determine user requirements and system specifications at the beginning of the system life cycle. Moreover, the requirements were not expected to change during the design, code, test, and deployment phases. In recent years it has been acknowledged that the time frame for identifying system requirements can vary depending on the life cycle model selected to guide development of the system. Some life cycle models assume that all requirements will be identified at the beginning of the project. Others provide for requirements to be identified in stages. They each present unique issues with respect to the analysis and specification of the user requirements and system specification.

3.1. Waterfall

One of the earliest, if not the earliest, life cycle model is the waterfall model of Royce. There are many variations on this model, which result in a wide range of distinct phases—from 3 to 20—from initial concept development to deployment and maintenance. The key characteristic of these models is the assumption that the opportunity for discovering requirements exists primarily at the beginning of the life cycle. There may be some iteration between the requirements analysis phase and the high-level design phase, but beyond that phase no significant changes in the requirements are made. This can mean that there will be no funds available later in the life cycle for review of requirements, for additional design, modifications to existing designs, test cases, draft user manuals, or other work products impacted by a change in requirements. The user must be aware that resources and funding cannot be added later in the process if the requirements prove to be inaccurate or incomplete. Changes in later phases of the development are a significant source of requirements creep, the insidious tendency of system requirements to change and add to the original cost and effort estimates.

Systems analysts must be aware of the impact of this type of model on their efforts to collect and understand the requirements for the system. Since it can take many forms and exist under many names, the systems analyst should examine the approach selected and assess it to determine if it is appropriate for the effort. Determining if it is appropriate means assessing the approach or life cycle model against the requirements effort to see if there will be sufficient time to identify user requirements and system specifications. Knowing if there will be sufficient time to identify requirements is not always easy. However, there are general guidelines that can help with the initial assessment.

The first thing to consider is whether the analyst has developed a similar system in the past. If the answer is yes, then there exists the possibility that the effort to collect requirements in the previous project can be used to estimate the effort required on the

current project. It may also tell the analyst whether it is feasible to uncover all the requirements at the beginning of the project or whether time should be set aside later in the development to revisit the requirements identification phase.

A second consideration is whether it is possible to know all the user requirements in the initial weeks and months of the project. If other user groups are to be included at later times, it is inevitable that additional requirements will be identified. Another consideration is whether all implementation components of the system will be known at the beginning of the project. If hardware is to be procured separately, particularly if it is an open competition in which more than one vendor is expected to compete, it will be impossible to complete the system specifications until that element is known.

Risk should also be considered in determining if it is feasible to complete the requirements and system specifications in the early weeks or months of the project. If, for example, there is risk that a chosen technical implementation alternative will not be available, time should be included to revisit and redefine the specification. This may be appropriate for performance requirements as well if new technology is to be included in the specification.

While the definition of user requirements and system specifications should avoid implementation details as much as possible, one or more of these considerations may be a practical aspect of real-world development projects and should be assessed for their impact on the identification of user requirements and system specifications. If the systems analyst determines that it is not feasible to identify significant sets of user requirements or system specifications during the early stages of the life cycle, then a type of phased implementation should be considered.

3.2. Alternative Life Cycle Models

In recent years the scope of system and software development projects has grown to encompass more and more functions and to involve larger numbers and types of entities, from command and control systems in automobiles to personal assistants and hand-held devices, and weather forecasting systems. In many of these systems it is difficult, if not impossible, adequately to define user requirements and system specifications early in the life cycle. In response to this, practitioners and academics have defined numerous development life cycles that allow iteration through the requirements phase at any point including after deployment and initiation of the maintenance cycle. Some names for these system development life cycles that may be encountered include incremental, phased, staged, operational prototype, object oriented, and variations on each. Each of these life cycle models offers additional opportunities to identify user requirements and system specifications, since they all assume that not all requirements are going to be known at the beginning of the project.

Some systems are undefined at the beginning because the domain is not well understood. For example, a system may be designed for one type of environment and it may be discovered that there are elements to that environment that were unaccounted for in the initial concept. For example, a form may be designed that requires entries be made in sequential order. When placed in the user environment, it may become obvious that the

ability to skip around is crucial to the usability of the form. Other systems may require functions that are not easily discerned at the start of the project. It may be well known, for example, that the environment for a robotic device will be extremely hot and dusty, but it may not be so quickly understood that an ability to sense dust accumulation and perform a clearing operation is possible and feasible. Still other systems may be intentionally addressed in subsets that are introduced over time.

Some life cycle models address specific issues. The spiral model, for example, addresses the numerous types of risk incurred in developing software-intensive systems. The spiral model introduces a series of prototypes that intentionally explore the risks—technical, managerial, or functional—that are anticipated during the system development. The objective of these life cycle models is to provide an additional opportunity for systems engineers to learn about the environment and domain for a system, so that its fit, form, and function are as good a match as is possible. The more that is known about the environment and domain in which a system will be built, the more probable it is that the appropriate life cycle and, therefore, the appropriate requirements elicitation and specification technique will be selected.

4. Methodologies and Techniques to Identify User Requirements and System Specifications

Life cycle models that support iteration through the requirements phase offer systems engineers the opportunity to employ a variety of techniques to uncover user requirements and system specifications. The specific objective of the requirements phase is to create a work product that can be used by the design team and test team to develop and test the system. Techniques employed to identify user requirements and system specifications should satisfy at least two objectives. The first objective is to uncover the user requirements and system specifications that are critical to a successful implementation. In addition, the same information needs to be identified for each requirement. For example, systems engineers should determine if they will need to know who wants each requirement, when it was identified, and whether it is mandatory, optional, or nice-to-have, for example. The second objective is to communicate the understanding of the users' needs to the users and to the design team. Since these groups are often quite different in their technical proficiency and expectations, this objective can offer significant challenges. Documentation of the user requirements and system specifications is also important to the control of changes to the system throughout its life.

-
-
-

Bibliography

Bahill T.A. and Dean F.F. (1999). Discovering system requirements. *Handbook of Systems Engineering and Management* (ed. A.P. Sage and W.B. Rouse), pp. 175–219. New York, N.Y.: John Wiley & Sons. [Presents the task of identifying system requirements from the perspective of the New Mexico Weapons Systems Engineering Center at Sandia National Laboratories. Section 4.8, "Related Items," presents a glossary of terms used in requirements elicitation and specification tasks.]

Office of the Under Secretary of Defense (Acquisition and Technology). (1996). *Department of Defense Guide to Integrated Product and Process Development, Version 1.0*, 41 pp. Washington, D.C.: Department of Defense. [Provides a primer for the defense acquisition workforce to foster, facilitate, and understand the use of integrated product and process development (IPPD). Its focus is how industry implements IPPD and how this impacts the Department of Defense's role in the acquisition process and the program office interfaces with their industrial counterparts.]

Office of the Under Secretary of Defense (Acquisition, Technology, and Logistics). (1999). *Rules of the Road: A Guide for Leading Successful Integrated Product Teams, Revision 1*, 41 pp. Washington, D.C.: Department of Defense. [Describes the IPT process for ACAT ID and ACAT IAM acquisition programs, but the concepts are applicable for all programs.]

Pressman R.S. (2000). *Software Engineering: A Practitioner's Approach, Fifth Edition*, 888 pp. New York, N.Y.: McGraw-Hill. [This provides a general discussion of software engineering topics and techniques, including methods for creation of the product, object-oriented and advanced topics, and management of the software engineering process.]

Quatrani T. (1999). *Visual Modeling with Rational Rose 2000 and UML*, 256 pp. Boston, MA: Addison-Wesley. [Provides a first look at how a process, a language, and a tool may be used to create a blueprint of a system.]

Reifer D.J. (2000). Requirements management: the search for Nirvana. *IEEE Software*, May/June, 45–47. [Describes difficulties associated with volatile requirements and how to manage them.]

Royce W.W. (1970). Managing the development of large software systems: concepts and techniques. *IEEE Report R–77-320* (Proceedings of WESCON, San Francisco, CA, August 1970), pp. 1–70.

Sage A.P. and Rouse W.B. (1999). An introduction to systems engineering and systems management. *Handbook of Systems Engineering and Management* (ed. A.P. Sage and W.B. Rouse), pp. 1–58. New York, N.Y.: John Wiley & Sons. [Provides a general overview of systems engineering and the major issues confronted by systems engineers in planning and managing systems engineering projects.]

Biographical Sketch

Ruth T. Buys is a VP at 3H Technology, LLC where she is responsible for corporate quality and process improvement. She led the successful effort to reach CMMI™ Maturity Level 3 and achieve ISO 90001:2000 certifications. Prior to 3H Technology, she was a Vice-President at CMS Information Services, Inc where she led its process improvement efforts resulting in a successful appraisal at CMMI™ Level 3 rating. Previous roles at CMS included supporting DoD/CIO office on software issues related to software best practices, measures and software standards policy. Prior to joining CMS Information Services, Ms Buys was a Research Associate Professor in the School for Information Technology and Engineering at George Mason University. There she taught decision support, application design and development, requirements definition, and object-oriented analysis and design methodology. At George Mason, she was a consultant to the federal government and private corporations on requirements analysis, configuration management, quality assurance, and project management, as well as process improvement. She has facilitated group meetings, including working sessions, and policy and planning activities using the automated group decision-support process Group Systems software at the US Army War College.Ms Buys has held technical and management positions at the MITRE Corporation, Booz, Allen and Hamilton, and other technology and research firms. She has taught graduate courses at Marymount and Drexel universities, including requirements analysis, comparative design methods and database management. Her PhD is from Drexel University in Information Science and Technology. In addition she has an MBA from George Washington University and her undergraduate degree is from the University of North Carolina at Chapel Hill. Ms Buys is a member of the Beta Phi Mu International Information Science Honor Society

and a member of INCOSE, ASQ, and IEEE. She has authored chapters for technical books, articles for refereed professional journals, and contributed to conference and workshop publications.