Chapter 15: System Requirements

Requirements are the foundation of an IT project. Failure to understand the requirements in the beginning results in an incorrect system and probably delays in delivery.

Requirements play a vital role in the systems development and maintenance processes. System requirements consist of hardware requirements, software requirements, and operational requirements. The IT project manager establishes a process to identify, define, elicit, and understand system requirements. The purpose is to establish a common understanding between the customers, users, stakeholders, and project manager of the requirements that will be completely addressed in the systems development. This chapter presents the importance of understanding system requirements.

System Requirement Identification

System requirement identification is important to help the customer and developers define and understand what will be involved in the system. The customer creates requirements for a specific purpose. These requirements are capabilities or conditions as stated by the customers, users, and stakeholders. Requirements can be functions, constraints, or other properties that must be provided, met, or satisfied so that the needs are filled for the systems intended users.

Requirements are the conditions that must be met for a system product to be acceptable to its customers, users, and stakeholders. Requirements can be totally new for an IT systems development project, or requirements can be for improving an existing IT system. This improvement can be possible by changing requirements in the existing system, enhancing the existing requirements, or correcting requirements to solve a problem in the existing system.

The customer and developers must understand the requirements before making a costly decision of what to build. This process involves determining, defining, and specifying requirements before analyzing them.

Requirement Determination

Requirement determination is a process that determines what is desired. Determining what is desired involves subprocesses, such as the customer defining the requirements and the system developer learning those requirements. The customer must state requirements clearly, rigorously, and precisely before proceeding to other system development phases.

The following questions are important in requirement determination:

- Who determines exactly what the requirements are?
- Does the customer know exactly what the requirements are?
- Does the IT project manager know exactly what the requirements are?
- Do the system developers know exactly what the requirements are?
- Do the system testers know exactly what the requirements are?

A combination of all of these people can determine the requirements. They form a team and work together to define, understand, and interpret requirements for a system as shown in Figure 15−1. Requirements are always unclear at the beginning. As time passes, the requirements mature because all of these members provide the necessary input. Each member of the team provides his or her input and makes the requirements clear and understandable.

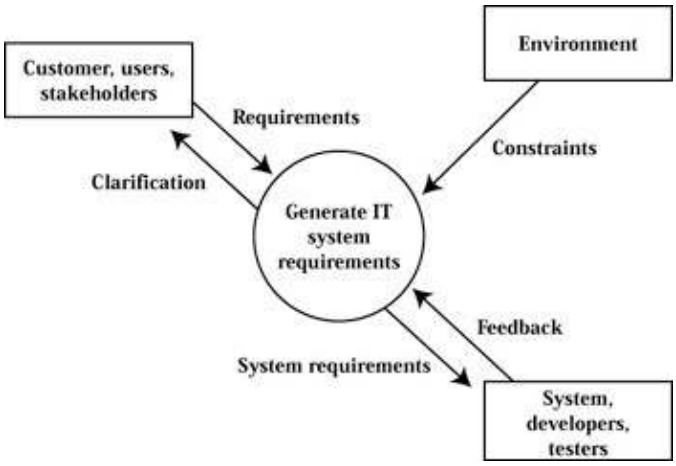

Figure 15−1: IT system requirements determination process

Requirement Elicitation

Requirement elicitation is a process performed by analysts who gather and understand information. This process involves the following factors:

- ***Fact−finding.*** Fact−finding uses mechanisms such as interviews, questionnaires, and observation of the operational environment of which the system will become a part.
- ***Validating the customers understanding of the gathered information.*** Validation involves creating a representation of elicitation results in a form that will focus attention on open issues that can be reviewed by those who provided the information. Possible representations include summary documents, usage scenarios, prototypes, and graphic models. A requirement proposal, requirement communication, and requirement definition achieve requirement elicitation. A requirement proposal outlines the need for customer requirements. The proposal requests an update that modifies or corrects an error in an existing system or develops a new system. A requirement proposal is sometimes referred to as a statement of work and states what is required from a system developer. It is an agreement that the customer and the system developer will abide by in the future. The requirement proposal may ask for the system developers ability to meet projected growth and operational changes. It can also ask for detailed costs, estimates of installation manpower, and an implementation plan. It may ask for a contractual agreement that will include responsibilities, delivery schedule, penalty clauses for missed deadlines, and explicit definitions of what constitutes the systems acceptance.
- ***Communicating open issues for resolution.*** Requirement communication involves the iterative elicitation and refinement of information from a variety of sources, which provides the customer with divergent perceptions of what is needed. The system analyst learns the applications and attributes of the product for delivery to the customer. The form of communication is meetings and phone conversations between the customer and the system analyst.

Frequent review of the evolving requirements by analysts who have domain expertise is essential to the convergence of this iterative process. The goal is to reach an agreement on the requirement definition. The result of the requirement definition is presented to a diverse audience, including system testers, for review and approval. Even though the customer and system analysts are frequently experts in their own areas, they are often less experienced in each others domain, which makes effective communication particularly difficult.

Requirement Definition

Requirement definition is a process. It is difficult to define requirements if they are not mature enough. The requirement may only be an idea in the customers mind. A customer must write explicitly his or her requirements. The primary function of defining requirements is to draw blueprints and document them to eliminate potential confusion and misinterpretation. Thus the requirement definition document that the customer produces will ensure that the system developers understand the customers requirements, needs, and objectives. This understanding and agreement will eliminate potential confusion and misinterpretation in the systems development.

Requirement definition usually includes an understanding of the environment in which the system can operate and how the system will interact with that environment. Explicit approval to proceed with requirement definition completes the elicitation process. The audience who must approve the requirements should agree that all relevant information sources have been contacted.

Importance of a Good Requirement

A good requirement is the foundation of the system. A good requirement is an agreement among the customers, users, stakeholders, and system developers. A study by the Standish Group in 1997 showed that American companies spent $100 billion for canceled software projects. Another $45 billion was spent on software projects that significantly exceeded their time and budget estimates. The Standish Group and other studies indicate the following top three reasons why software projects fail:

1. Requirements and specifications are incomplete.
2. Requirements and specifications are changing too often.
3. The project has a lack of user input.

A good requirement should use imperative phrases in the requirement specification. Imperative phrases command that something must be provided.

- Shall means prescribes and is used to dictate the provision of a functional capability.
- Will means describes and is used to cite things that the operational or developmental environments are to provide to the capability being specified.
- Must and must not indicate constraints. Must is often used to establish performance requirements or constraints.
- Should means suggest and is not used as an imperative in requirement specification statements.

Samples of Good, Bad, and Ugly Requirements

Good Requirements

- The organizations web site shall provide the customers and the public with accurate, timely, and relevant information on the missions and functions of the organization.
- The web site shall contain a clearly defined purpose that supports the mission of the organization and achievements.
- The web site shall be developed in accordance with the organizations management policy.
- The web site shall be maintained continuously with current data, updated at least biweekly.
- The point of contact for this requirement shall be a member of the organizations web site working

group and performs tasks as directed by the working group chairperson.
- The policy, procedure, and standard are established and delivered to the web site working group chairperson not later than 30 days before the activation of the organizations web site.
- Acceptance criteria: The web sites and pages shall be user friendly, clear, and concise. The organizations graphic representations shall be color matched and easy to understand, and the response time shall be less than 5 seconds per page. The project will be completed in 6 months. The cost allocated for this project is $100,000, and allocated manpower is three professionals.

Bad Requirements

- The organizations web site shall provide the customers and the public with accurate, timely, and relevant information on the missions and functions of the organization.
- The web site shall contain a clearly defined purpose that supports the mission of the organization and achievements.
- The web site shall be developed in accordance with the organizations management policy.
- The web site shall be maintained continuously with current data.

Ugly Requirements

The organizations web site shall provide the customers and the public with accurate, timely, and relevant information on the missions and functions of the organization.

Requirement Types

The requirement types are functional and nonfunctional. Functional requirements are the capabilities that the system will perform. An example of a functional requirement is formatting of text. Nonfunctional requirements are the constraint on the system. Nonfunctional requirements are the performance, maintainability, safety, and reliability requirements. In addition, requirement types are categorized as follows:

- ***Derived.*** Derived requirements are deduced or inferred from the collection and organization of requirements into a particular systems configuration and solution.

  Example: A user needs a communication system with certain operational requirements. The acquirer adds requirements concerning interoperability with other systems.
- ***Explicit.*** Explicit requirements are written clearly.

  Example: An armored personnel carrier must be capable of sustaining a velocity of 40 kph over level ground for not less than 4 hours while carrying a full complement of troops and gear.
- ***Decomposed.*** Decomposed requirements clearly explain the task, action, or activity that must be accomplished to achieve a desired outcome.

  Example: A fighter aircraft must be able to fly a certain distance under normal conditions before refueling. This yields requirements about fuel tank capacity, airframe configuration, expected cruising speed and altitude (which will yield further requirements for the human−system interface), aircraft weight, and so on.
- ***Implied.*** Implied requirements explain that the requirements are not stated but include hidden meaning.

  Example: There is no benefit to be gained from buying a highly superior computer if it cannot exchange information with the computers already incorporated within the infrastructure.

Characteristics of a Good Requirement Specification

The characteristics of a good requirement are as follows:

- The customer requirements have been specified uniquely
- No duplication or overlapping of the requirements occurs.
- The requirements are needed for further analysis.
- The requirements are feasible and implementable and are not outside the capability of current technology.
- The requirements have been stated.

The characteristics of a good requirement specification are as follows:

- Unambiguous
- Complete
- Verifiable
- Consistent
- Correct
- Modifiable
- Traceable
- Testable
- Usable after development

These characteristics are interrelated. For example, requirement completeness and consistency are factors that contribute to correctness. Modifiability and testability are factors that contribute to maintainability. A requirement specification is not correct if it is incomplete or inconsistent. It is difficult to validate a requirement specification that is not understood.

Unambiguous

Only one interpretation of every stated requirement should exist. The many ambiguities in natural language must be guarded against, and ambiguity can be avoided by using a formal requirement specification language. In an object−oriented approach, a requirement is defined as follows:

- ***Object.*** Object−based requirements are organized in terms of real−world objects, their attributes, and the services performed by those objects.
- ***Process.*** Process−based approaches organize the requirements into hierarchies of functions that communicate via data flows.
- ***Behavior.*** Behavioral approaches describe the external behavior of the system in terms of abstract notion, mathematic functions, and state machines.

Complete

Every significant requirement that is concerned with system function, performance, attributes, interfaces, and design constraints should be included. Completeness also requires conformity with specified standards. The terms and units used should be defined, and all sections of the specification should be fully labeled. Phrases like to be determined (TBD), infinite quantity, maximum, and minimum should be avoided.

Verifiable

Requirements can be verified if they are stated in concrete or absolute terms and involve measurable quantities. All requirements except the nonfunctional requirements shall be verifiable.

Consistent

Stated requirements should not use conflicting terminology when referring to the same system element or call for different forms of logical or data structure. For example, the format of an output report may be described in one requirement as tabular but in another as textual. One requirement may state that all lights shall be red while another states that all lights shall be yellow. One requirement may specify that the program will add two inputs and another may specify that the program will multiply them. One requirement may state that B must always follow C, whereas another requires that B and C occur simultaneously.

Correct

Requirements should be defined unambiguously, clear, understandable, and in a correct logical natural language. Each requirement should be unique and consist of one meaning only. The customer has given consideration to each requirement and clarifies for hidden assumptions.

Modifiable

The specification should be structured and annotated so that changes to requirements can be made easily, completely, and consistently. The system requirement specification (SysRS) should have a coherent and easy−to−use organization with a table of contents, an index, and explicit cross−references. The same requirement should not appear in more than one place in the SysRS. Each requirement is expressed separately rather than intermixed with other requirements.

Traceable

The origin of each requirement should be clear, and the format of the specification should facilitate referencing forward and backward. Forward traceability depends on each requirement in the SysRS having a unique name and reference number. Backward traceability depends on each requirement explicitly referencing its source in earlier documents. For example, hierarchic numbers (1.2.3. 1.2.3.4, 1.2.3.4.5.6.7.8) are useful in the requirement traceability. Each requirement can be traced from its inception to its deployment in the delivered system forward and backward. The customers requirement can be traced to its related system requirement, test requirement, designs, and program or component module. This allows impact analysis to be conducted easily any time a change to a requirement is identified or anticipated. Once the impact of the change has been identified, the customer immediately knows what components related to the changed requirement need to be changed or replaced.

Testable

All requirements in the specification should be well defined so that they can be easily tested and validated. Each requirement can have its own verification and validation criteria. This provides a high level of quality for each requirement that is propagated across the entire systems development process. It also allows the testing process to be initiated earlier in the systems development life cycle.

Usable after Development

After the system goes into operation, it should be possible to use the specification for identification of maintenance requirements and constraints.

System Requirement Specification

SysRS contains operational, technical, and logistic capabilities that are required in the IT system. The SysRS document includes the identification, organization, presentation, and modification of the requirements. The SysRS document incorporates the systems operational concepts, design constraints, and design configuration requirements into the specification. The SysRS document includes the following for the IT system:

- Desired operational capabilities
- Physical characteristics
- Performance parameters and expected values
- Interfaces with its environment
- Functional requirements
- Interactions with its environment
- Documentation requirements
- Reliability requirements
- Logistic requirements
- Constraint requirements
- Personnel requirements
- Quality requirements
- Security requirements
- Safety requirements
- Training requirements

In addition, the SysRS contains necessary characteristics and qualities of individual requirements and the set of all requirements for the system as follows:

- ***Unique set.*** Each requirement shall be stated only once.
- ***Normalized.*** Requirements shall not overlap (i.e., they shall not refer to other requirements or the capabilities of other requirements).
- ***Linked set.*** Explicit relationships shall be defined among individual requirements to show how the requirements are related to form a complete IT system.
- ***Complete.*** The SysRS shall include all of the requirements identified by the customer and those needed for the definition of the IT system.
- ***Consistent.*** The SysRS content shall be consistent and noncontradictory in the level of detail and style of requirement statements.
- ***Bounded.*** The boundaries, scope, and context for the set of requirements shall be identified.
- ***Modifiable.*** The SysRS shall be modifiable. Clarity and nonoverlapping requirements shall be avoided.

The objective of the SysRS is to define a capability that satisfies a customers statement of work as shown in Figure 15−2. The purpose of the SysRS is to provide a medium for capture of what is needed and communication to all interested parties. The SysRS document serves as a contract between the customer and the system developers by defining what is to be provided and, in many instances, the manner in which it is to be produced and the technology that it incorporates. In addition, the SysRS provides a basis for most of the projects management and engineering functions, such as assessing proposed engineering changes, resolving

acquirer and provider disputes, developing test requirements, writing the preliminary users manual, planning maintenance support activities, and implementing operational enhancements.

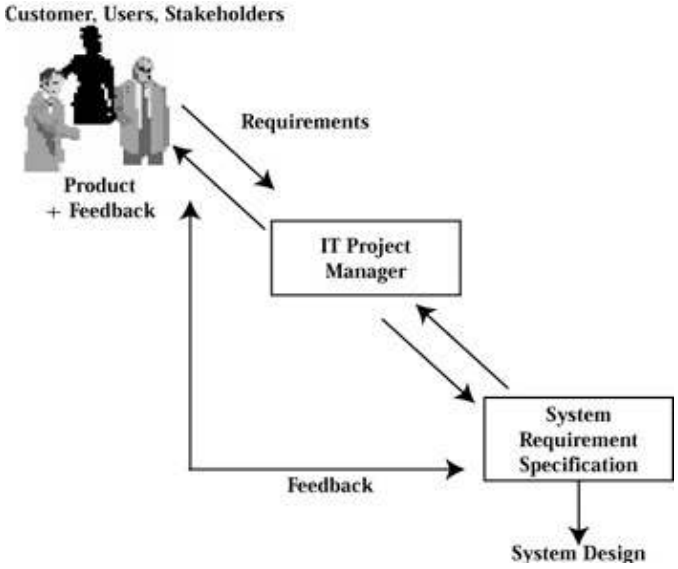

Figure 15−2: IT system requirement specification

For the SysRS to serve these purposes, it must provide a definition of the future operational and support environments in which the required capability will exist and provide a prescription for the system intended as the solution to the customers statement of work (SOW). If necessary the SysRS must also identify the methodologies and technologies that are to be used during the systems development life cycle.

The SysRS must describe the strategic and tactical environments and the systems solution within those environments. The descriptions of these environments must address elements of the operational and support situations that will impinge on the solution as they are expected to exist during the operational lifetime of the systems solution. The customer and developers must distinguish between those aspects of the environment (strategic, tactical, operational, and support) that are continuations of the existing situation and those that are expected to be delivered.

The design of the SysRS document should be based on a predetermined approach to structuring the specifications information into sections and subparagraphs (Box 15−1). Statements that are part of a single function are normally grouped together. Functions that are connected in series by output to input relationships should appear in the document in that same sequence if possible. Functions that share common inputs and outputs should be addressed within the same area of the document. If several processes must be accomplished in the same time frame, the document must tie their specifications together to make this commonality clear. Functions that are similar need to be distinguished from one another.

Box 15−1: SysRS Outline

---

1. Introduction

    1.1 Systems purpose

    1.2 Systems scope
2. Systems description

    2.1 Systems context

2.2 Assumptions and dependencies

2.3 Users characteristics

2.4 Operational scenarios

3. Systems interfaces
4. Systems capabilities, conditions, and constraints

4.1 Physical

4.2 Systems performance characteristics

4.3 Systems safety and security

4.4 Systems operation

4.4.1 Systems human factors

4.4.2 Systems maintainability

4.4.3 Systems reliability

5. Systems external interfaces
6. Systems testing scenarios
7. Systems training plan
8. Definitions, acronyms, and abbreviations
9. References

Data item descriptions (DIDs) that prescribe the SysRS outline and content are attempts to solve the structural problem for a general class of documents. These DIDs only resolve issues at the highest level of organization. The DID that is to govern a particular SysRS must be tailored to fit the type of project that is to use it and the category of system that is to be developed. The parameter values shall be clearly and uniquely defined. The SysRS shall not include the values that cannot be tested, such as the following:

- TBD
- Infinity
- As a minimum, maximum
- Not limited to
- As appropriate
- Equivalent to
- If practical
- Large, small
- Many, most
- Timely
- Easy
- Normal
- Adequate

The sentences in SysRS include words and phrases that are simple and appropriate to the intent of the statement. For example, the word obscure is not used if hide is meant.

Preparation of Test Plans, Procedures, and Test Cases

Preparation of test plans, procedures, and test cases is critical in the implementation of IT systems. The requirement testing plan begins at the first phase of the systems development in which the correction of errors is least costly and the largest portion of bugs have their root cause. Identification of requirement defects at the earlier phase improves the quality of requirements. The presence of inadequate requirements is often the reason for a projects failure.

The testing starts early, when the customer initiates requirements for a system. The SysRS is the first document that introduces the systems testing. Users provide test cases to test the system; testers cannot test a system if the customer does not provide the test cases. These requirements are documented in the SysRS. The SysRS is integrated throughout the systems development life cycle.

Automated test tools are available that assess the effectiveness of testing a system. Proper implementation of a database tracks requirements at each level of decomposition and tests associated with the verification of these requirements. From this database the project can gain insight into the relationship between the test and the customers requirements.

Case Study 15−1

From humble beginnings to a multimillion dollar dietary supplement company, Mannatech lent credence to its founders belief that their network−marketing venture was something with phenomenal promise. In just 5 years the company has become a leader in what is termed the nutraceutical industry. Mannatech has a staff of roughly 300 people supporting a North American force of over 400,000 home−based associates committed to building consumer demand for Mannatechs proprietary, natural supplements. With a diverse network of nutritional experts and yearly revenues topping $152 million, Mannatech has begun expanding into international markets.

According to Bill Councill, Mannatechs Systems and Process Engineering Manager, the companys growth can be attributed to its unique commission system. We pay eight different types of commissions, which differentiates us from other markets and certainly from other network marketing companies. Mannatechs software system manages the complex algorithms that allow for the movement of commissions throughout the associate sales structure, called the *down line.* Councill emphasizes the importance of a technologically astute software system to support such a network. It is important that our software does not have any downtime so that products are distributed promptly and commissions are paid accurately throughout the network. Our internal employees and associates will not tolerate software that does not work.

Given its fast growth and global expansion, the company has taken prudent steps toward even better management of its technological infrastructure, implementing highly dependable software applications with robust Internet capabilities. Such software capabilities enable Mannatech to support a complex commission plan for its associates and maintain a continuous flow of products and materials in a timely and accurate fashion domestically and internationally.

In many ways, development of nutritional products that would help Mannatech rise to the top of the nutraceutical market was the easy part. Developing requirements−driven applications to support the network of associates, maintaining communication between distributed groups, and meeting end−user specifications, on time and within budget, was the real challenge.

Before requirements management, Mannatech did not use a formal requirement−management process or tools for delivering software applications; rather they depended on functional specification manuals. For every

month we were over time on one particular software development project, it was costing us nearly half a million dollars, explained Éoin Redmond, Vice President of Information Technology for Mannatech. We had no way of doing traceability of requirements. We had no way to easily modify or amend requirements or effectively communicate those changes. It created an atmosphere that requirements werent important after all and that we would simply get a general idea of users requirements and figure it out as we went along.

With no formal process for defining, managing, and changing requirements, Mannatechs development teams found themselves in a code−and−go environment, bearing the burden of a projects specifications. Development time lines were simply speculative guesses, and as time went on, development costs accrued, going far beyond the projected costs. When the project was handed over to the quality assurance (QA) team, the lack of traceability between requirements and tests resulted in unfocused or incomplete testing.

Such a code−and−go approach yielded low−quality systems, which dissatisfied end−users and dampened the developers morale. Trust in the project team had been compromised, and the end−users held the team responsible. The project team spent months attempting to resolve issues that were a direct result of unclear, undefined, and unauthorized changes to requirements.

Mannatechs project managers realized that it was time for a change. Redmond said, I am a very firm believer that if you dont control your requirements, then the rest of the project is always going to have problems. The project managers decided that a formal requirement definition and management process was the first step toward improving the quality and user acceptance of their products. They needed a way to trace requirements to specific use cases and tests to ensure a requirement−driven rather than an ad−hoc approach to development and testing.

With plans to open their first international office in Australia, requirement management became the top priority during the development of that offices software system. Mannatech required a powerful tool that would aid in communication of the software development process, providing the following:

- An intuitive, easy−to use interface for nontechnical and technical end−users
- Communication among all departments involved in the development life cycle
- A Web−based component to form a tightly knit integration between corporate and international subsidiaries and to transmit various financials
- Mannatechs proprietary iterative and incremental development cycle, referred to as MannaFacture^{tm}, demanded tools that could support and improve their efforts without training and upkeep. From a technical standpoint, the project managers required the following:
- A robust architecture that was flexible and scalable enough to support current and future development projects
- Customizable options that were within the scope of the tool itself and through integration with custom and third−party tools for design and testing
- A framework−based tool to support the expansion of the products functionality and scope
- A product with an established user group from which the project managers could gain insight into additional uses for the tool

The project managers needed to integrate requirements to related development and testing information. As the company moved forward with plans to achieve a successful Software Engineering Institutes (SEI) capability maturity model (CMM) review, such an integrated solution would greatly increase their ability to meet, if not exceed, SEIs high standards of quality. Mannatech sought a vendor with whom they could form a partnership and successfully achieve the goals that they had outlined.

As Mannatech evaluated various automated requirement management tools for the Australia opening, Caliber−RM^{tm} from Technology Builders^{tm}, Inc., consistently met their standards. Caliber−RM^{tm} offers a

collaborative environment for requirement management, including a group discussion feature and Web interface that is critical for involving end−users from the various departments and Mannatechs international subsidiaries in the ongoing requirements process. Through its Software Quality Management (SQM) Framework[tm], Caliber−RM[tm] provided the expansion capabilities that Mannatech needed to integrate requirement management and test management and object modeling tools.

Caliber−RM[tm] is integrated with Mercury Interactives TestDirector, an automated test management tool, and SELECT Software Tools (SELECT Enterprise), an object−modeling tool, both of which Mannatech already had implemented. Through these integrations, Mannatech has the ability to trace requirements to use cases and tests to ensure that the development and testing are driven by the actual requirements of the application. In addition, the SQM Framework provides the means to integrate additional tools, including problem/change request tracking, risk management, project management, and code coverage (from Technology Builders, Inc., 400 Interstate North Pkwy, Suite 1090, Atlanta, GA, 30339).

System Requirement Checklist

The IT project manager should check for the following types of requirements in the SysRS specification:

- Acceptance testing
- Data handling
- Design standards
- Maintainability
- Reliability
- Reusability
- Timing and sizing