

Lecture 01

The Importance of Requirements

The purpose of this book is to help you improve the practice of requirements engineering. Requirements engineering is difficult. It's not just a simple matter of writing down what the customer says he wants. A fundamental problem in business is that requirements are inherently dynamic; they will change over time as our understanding of the problem we are trying to solve changes. The importance of good requirements and the underlying dynamic nature of the process mean that we must be as accurate as possible, and yet be flexible. Flexible does not mean "weak," but rather than we have a *process* for developing requirements and accommodating changed requirements as we clarify the real requirements of customers. Ineffective requirements practices are an industrywide problem. This is an area in which you can have a major positive impact. A more disciplined approach to requirements development and management is needed in order to improve project success rates. An alarming 53% of industry's investment in technical development projects is a casualty of cost overruns and failed projects.

This chapter defines the term "requirement," explains why requirements are important, and advocates planning to define the requirements strategy and activities. It suggests use of a defined and documented requirements process, that investing more in the requirements process will have a large payback, and that requirements serve a crucial role in business in managing risk. It recommends that you consider certain factors in making your career decisions. It suggests that much of the advice provided in the book is applicable to projects of all sizes.

What Are Requirements and Why Are They Important?

A *requirement* is a necessary attribute in a *system*, a statement that identifies a capability, characteristic, or quality factor of a system in order for it to have value and utility to a *customer or user*. Requirements are important because they provide the basis for all of the development work that follows. Once the requirements are set, developers initiate the other technical work: system design, development, testing, implementation, and operation. Too often, there is a tendency to want to start what is often referred to as "the real work" (developing, or programming, the software) too quickly. Many customers and project managers (PMs) seem to believe that actual programming work ("coding") indicates that progress is being made. According to industry experience, insufficient time and effort are spent on the requirements-related activities associated with system development. Industry experience confirms that a better approach is to invest more time in requirements gathering, analysis, and management activities. The reason is that, typically, coding work is started much sooner than it should be because additional time is needed to identify the "*real*" requirements and to plan for requirements-related activities (described below).

There is a significant difference between “*stated*” requirements and “real” requirements. Stated requirements are those provided by a customer at the beginning of a system or software development effort, for example, in a request for information, proposal, or quote or in a statement of work (SOW). Real requirements are those that reflect the verified needs of users for a particular system or capability. There is often a huge difference between the stated requirements and the real requirements. Analysis of the stated requirements is required to determine and refine real customer and user needs and expectations of the delivered system. The requirements need to be filtered by a process of clarification of their meaning and identification of other aspects that need to be considered. To cite a simple example, *requirements analysts* (RAs) are more familiar with the need to state requirements clearly (see the criteria for a good requirement provided below). There are many ways in which the capability, understanding, and communication of the meaning of each and every requirement may be different to a user than to a developer. Therefore, it is vital (and time saving) that all requirements be clarified through the mechanism of a joint customer/user and RA effort. Customers and users need the support of technically trained and experienced professionals, and vice versa, to ensure effective communication. Developers need to have that same understanding so that the solution they define addresses the needs in the way everyone expects. Misunderstandings of requirements result in wasted effort and rework. Another important insight is that sometimes the requirements are *unknowable* at the outset of a development effort because they are affected by the new capabilities to be provided in the new system. This suggests the need to plan for new and changed requirements—to provide a degree of flexibility.

Identifying the real requirements requires an *interactive and iterative* requirements process, supported by effective practices, processes, mechanisms, methods, techniques, and tools. This book provides a description of *how* the RA can use these in performing the needed work. In a previous book, *Effective Requirements Practices* [1], I describe *what* should be done and provide an extensive set of references to many of the best publications in the industry literature. This book is intended to provide a concise handbook that serves as a desk reference guide for the RA or engineer and requirements manager in engineering and computing. It also provides updated references. The requirements process need not be complicated or expensive. However, a requirements process is *required* for a project of any size. It’s most important that a project or organization *have* a defined and documented requirements process. The nature of the specific components of the defined process can be improved based on experience.

Why Plan?

It’s well known and understood by most people that a bit of planning goes a long way. For example, before leaving on an automobile trip, checking a map to locate the destination and, perhaps, even planning a route may be time well spent. Yet, we frequently charge ahead with the doing with little or no planning, don’t we? It’s human nature to want to get on with the needed work without doing much planning.

Systems development and software development managers and practitioners are familiar with several types of plans: project plan, systems engineering management plan (SEMP), quality assurance (QA) plan, configuration management (CM) plan, software development plan (SDP), test plan, and so on. However, the concept of a *requirements plan* may be new to you. Leveraging requirements-related activities has great power and effect. Writing a requirements plan maximizes value. A requirements plan defines how the real requirements will evolve and how the requirements activities will be addressed.

Writing a requirements plan (RP) facilitates an understanding of the activities and efforts that need to be undertaken to implement an effective requirements process for a particular development effort. Additional details concerning the requirements plan are provided below.

A Suggested Strategy

I suggest a strategy that includes (1) writing a requirements plan, (2) designing or tailoring a requirements process for your project, (3) investing in the requirements-related activities in the system life cycle, and (4) utilizing the effective requirements practices, mechanisms, methods, techniques, tools, and training that are described in this book.

Requirements Activities in the System Life Cycle

Managers often think of requirements-related activities as consisting primarily of gathering requirements and managing changes to those requirements throughout the life cycle. In reality, there are several other requirements-related activities that need to be addressed in the *system life cycle*:

Identifying the stakeholders: This includes anyone who has an interest in the system or in its possessing qualities that meet particular needs.

Gaining an understanding of the customers' and users' needs for the planned system and their expectations of it: This is often referred to as *requirements elicitation*. Note that the requirements can include several types. Types of requirements are discussed in Chapter 4. Requirements gathering techniques are discussed in Chapter 5.

Identifying requirements: This involves stating requirements in simple sentences and providing them as a set. Business needs or requirements are the essential activities of an enterprise. They are derived from business goals (the objectives of the enterprise). Business *scenarios* may be used as a technique for understanding business requirements. A key factor in the success of a system is the extent to which it supports the business requirements and facilitates an organization in achieving them

.
Clarifying and restating the requirements: This is done to ensure that they describe the customer's real needs and are in a form that can be understood and used by developers of the system.

Analyzing the requirements: This is done to ensure that they are well defined and that they conform to the criteria of a good requirement (provided below).

Defining the requirements in a way that means the same thing to all of the stakeholders: Note that each stakeholder group may have a significantly different perspective of the system and the system's requirements. Sometimes this requires investing significant time learning a special vocabulary or project lexicon. Often it requires spending considerable time and effort to achieve a common understanding.

Specifying the requirements: This requires including all of the precise detail of each requirement so that it can be included in a specification document or other documentation, depending on the size of the project.

Prioritizing the requirements: All requirements are not of equal importance to the customers and users of the planned system. Some are critical, some of relatively high priority, some of normal or average priority, and some even of lower priority. It is important to prioritize all of the requirements because there is never enough time or money to do *everything* we'd like to do in our developed systems. Prioritizing the requirements provides the opportunity to address the highest priority first and possibly release a version of a product later that addresses

lower-priority needs. Prioritizing helps ensure that an appropriate amount of investment is made in meeting various customer needs.¹

Deriving requirements: There are some requirements that come about because of the design of a system, but do not provide a direct benefit to the end user. A requirement for disc storage might result from the need to store a lot of data, for example.

Partitioning requirements: We categorize requirements as those that can be met by hardware, software, training, and documentation, for example. Often this process turns out to be more complex than we anticipate when some requirements are satisfied by more than one category.

Allocating requirements: We allocate requirements to different subsystems and components of the system. The allocations may not always be satisfied by just one subsystem or component.

Tracking requirements: We need the capability to trace or track where in the system each requirement is satisfied, so that we can verify that each requirement is being addressed. This is most often accomplished through use of an automated requirements tool.

Managing requirements: We need to be able to add, delete, and modify

requirements during all of the phases of system design, development, integration, testing, deployment, and operation. The *requirements repository* consists of a set of artifacts and databases. It is described in Chapter 5.

Testing and verifying requirements: This is the process of checking requirements, designs, code, test plans, and system products to ensure that the requirements are met.

Validating requirements: This is the process for confirming that the real requirements are implemented in the delivered system. The order of validation of requirements should be prioritized since there is a limited amount of funding available.

