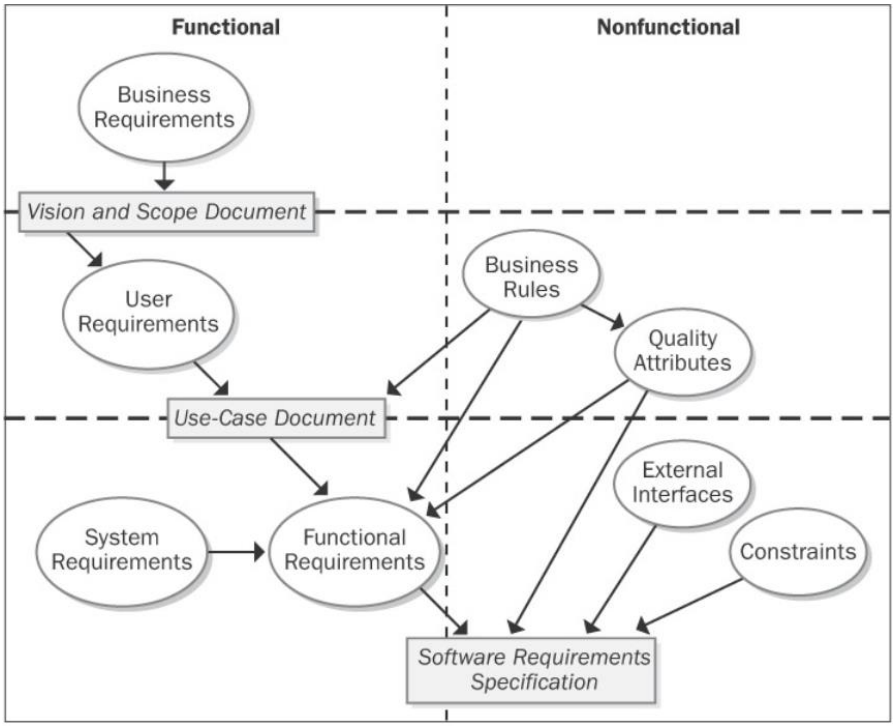
Software Requirements

Some Interpretations of Requirement

1. A condition or capability needed by an user to solve a problem or achieve an objective
2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.
3. A documented representation of a condition or capability as in 1 or 2.

Levels of Requirements



Business requirements represent high-level objectives of the organization or customer who requests the system.

User requirements describe user goals or tasks that the users must be able to perform with the product.

Functional requirements specify the software functionality that the developers must build into the product to enable users to accomplish their tasks, thereby satisfying the business requirements.

The term *system requirements* describes the top-level requirements for a product that contains multiple subsystems—that is, a *system* (IEEE 1998c).

Business rules include corporate policies, government regulations, industry standards, accounting practices, and computational algorithms.

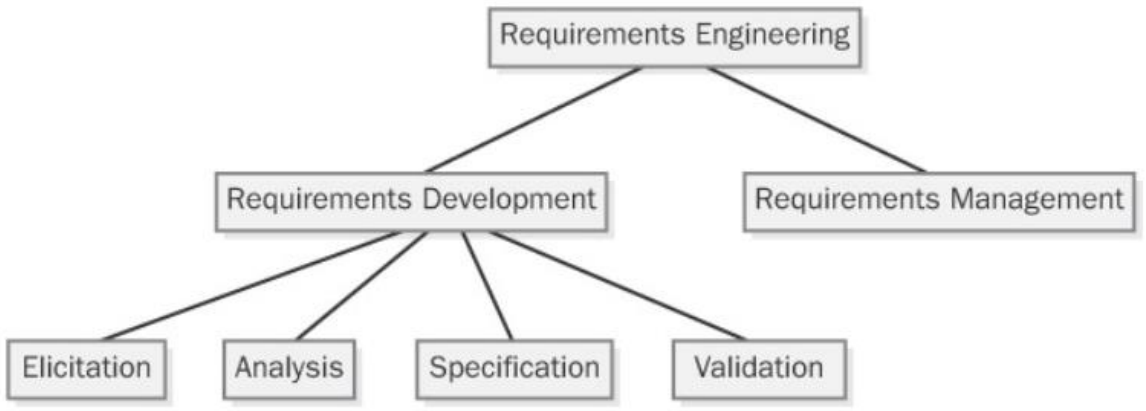
Functional requirements are documented in a *software requirements specification* (SRS), which describes as fully as necessary the expected behavior of the software system.

Quality attributes augment the description of the product's functionality by describing the product's characteristics in various dimensions that are important either to users or to developers.

Constraints impose restrictions on the choices available to the developer for design and construction of the product.

A *feature* is a set of logically related functional requirements that provides a capability to the user and enables the satisfaction of a business objective.

What Requirements Are Not



Requirements Development

They include the following:

- Identifying the product's expected user classes
- Eliciting needs from individuals who represent each user class
- Understanding user tasks and goals and the business objectives with which those tasks align

Requirements Management

Requirements management entails "establishing and maintaining an agreement with the customer on the requirements for the software project"

They include the following:

- Defining the requirements baseline (a snapshot in time representing the currently agreed-upon body of requirements for a specific release)
- Reviewing proposed requirements changes and evaluating the likely impact of each change before approving it

- Incorporating approved requirements changes into the project in a controlled way

Every Project Has Requirements

Every software application or software-containing system has users who rely on it to enhance their lives. The time spent understanding user needs is a high-leverage investment in project success

When Bad requirements Happen to Nice people

Shortcomings in requirements practices pose many risks to project success, where *success* means delivering a product that satisfies the user's functional and quality expectations at agreed-on cost and schedule.

Insufficient User Involvement

Insufficient user involvement leads to late-breaking requirements that delay project completion.

Creeping User Requirements

To manage scope creep, begin with a clear statement of the project's business objectives, strategic vision, scope, limitations, success criteria, and expected product usage. Evaluate all proposed new features or requirements changes against this reference framework.

Ambiguous Requirements

Ambiguity leads to different expectations on the part of various stakeholders. Some of them are then surprised with whatever is delivered. Ambiguous requirements result in wasted time when developers implement a solution for the wrong problem. Testers who expect the product to behave differently from what the developers intended waste time resolving the differences.

Gold Plating

Gold plating takes place when a developer adds functionality that wasn't in the requirements specification but that the developer believes "the users are just going to love." The use-case approach for eliciting requirements helps to focus requirements elicitation on the functionality that lets users perform their business tasks.

Minimal Specification

Sometimes marketing staff or managers are tempted to create a limited specification, perhaps just a product concept sketched on a napkin. They expect the developers to flesh out the spec while the project progresses. This might work for a tightly integrated team that's building a small system, on exploratory or research projects, or when the requirements truly are flexible (McConnell 1996). In most cases, though, it frustrates the developers (who might be operating under incorrect assumptions and with limited direction) and disappoints the customers (who don't get the product they envisioned).

Overlooked User Classes

Most products have several groups of users who might use different subsets of features, have different frequencies of use, or have varying experience levels. If you don't identify the important user classes for your product early on, some user needs won't be met. After identifying all user classes, make sure that each has a voice.

Inaccurate Planning

Vague, poorly understood requirements lead to overly optimistic estimates, which come back to haunt us when the inevitable overruns occur. An estimator's off-the-cuff guess sounds a lot like a commitment to the listener. The top contributors to poor software cost estimation are frequent requirements changes, missing requirements, insufficient communication with users, poor specification of requirements, and insufficient requirements analysis

Benefits from a High-Quality Requirements Process

- Fewer requirements defects
- Reduced development rework
- Fewer unnecessary features

Characteristics of Excellent Requirements

Requirement Statement Characteristics

- Complete
- Correct
- Feasible
- Necessary
- Prioritized
- Unambiguous
- Verifiable

Requirement Statement Characteristics

- Complete
- Consistent
- Modifiable
- Traceable

Table 3-1: Requirements Engineering Good Practices

Knowledge	Requirements Management	Project Management
▪ Train requirements analysts ▪ Educate user reps and managers about requirements ▪ Train developers in application domain ▪ Create a glossary	▪ Define change-control process ▪ Establish change control board ▪ Perform change impact analysis ▪ Baseline and control versions of requirements ▪ Maintain change history ▪ Track requirements status ▪ Measure requirements volatility ▪ Use a requirements management tool ▪ Create requirements traceability matrix	▪ Select appropriate life cycle ▪ Base plans on requirements ▪ Renegotiate commitments ▪ Manage requirements risks ▪ Track requirements effort ▪ Review past lessons learned

Requirements Development

Elicitation	*Analysis*	*Specification*	*Validation*
▪ Define requirements development process ▪ Define vision and scope ▪ Identify user classes ▪ Select product champions ▪ Establish focus groups ▪ Identify use cases ▪ Identify system events and responses ▪ Hold facilitated elicitation workshops ▪ Observe users performing their jobs ▪ Examine problem reports ▪ Reuse requirements	▪ Draw context diagram ▪ Create prototypes ▪ Analyze feasibility ▪ Prioritize requirements ▪ Model the requirements ▪ Create a data dictionary ▪ Allocate requirements to subsystems ▪ Apply Quality Function Deployment	▪ Adopt SRS template ▪ Identify sources of requirements ▪ Uniquely label each requirement ▪ Record business rules ▪ Specify quality attributes	▪ Inspect requirements documents ▪ Test the requirements ▪ Define acceptance criteria

Knowledge

- Train requirements analysts
- Educate user representatives and managers about software
- Train developers in application domain concepts
- Create a project glossary

Requirements Elicitation

- Define a requirements development process
- Write a vision and scope document
- Identify user classes and their characteristics
- Select a product champion for each user class

Requirements Analysis

- Draw a context diagram
- Create user interface and technical prototypes
- Analyze requirement feasibility

Requirements Specification

- Adopt an SRS template
- Identify sources of requirements
- Uniquely label each document

Requirements Validation

- Inspect requirements documents.
- Test the requirements
- Define acceptance criteria

Requirements Management

- Define a requirements change-control process
- Establish a change control board
- Perform requirements-change impact analysis.

Project Management

- Select an appropriate software development life cycle.
- Base project plans on requirements.
- Renegotiate project commitments when requirements change.

Getting Started with New Practices

Table 3-2: Implementing Requirements Engineering Good Practices

Impact			Difficulty
	High	***Medium***	***Low***
High	▪ Define requirements development process ▪ Base plans on requirements ▪ Renegotiate commitments	▪ Identify use cases ▪ Specify quality attributes ▪ Prioritize requirements ▪ Adopt SRS template ▪ Define change-control process ▪ Establish CCB ▪ Inspect requirements documents ▪ Allocate requirements to subsystems ▪ Record business rules	▪ Train developers in application domain ▪ Define vision and scope ▪ Identify user classes ▪ Draw context diagram ▪ Identify sources of requirements ▪ Baseline and control versions of requirements
Medium	▪ Educate user reps and managers about requirements ▪ Model the requirements ▪ Manage requirements risks ▪ Use a requirements management tool ▪ Create requirements traceability matrix ▪ Hold facilitated elicitation workshops	▪ Train requirements analysts ▪ Select product champions ▪ Establish focus groups ▪ Create prototypes ▪ Define acceptance criteria ▪ Perform change impact analysis ▪ Select appropriate life cycle	▪ Analyze feasibility ▪ Create a glossary ▪ Create a data dictionary ▪ Observe users performing their jobs ▪ Identify system events and responses ▪ Uniquely label each requirement ▪ Test the requirements ▪ Track requirements status ▪ Review past lessons learned
Low	▪ Reuse requirements ▪ Apply Quality Function Deployment ▪ Maintain change history volatility	▪ Measure requirements ▪ Track requirements effort	▪ Examine problem reports

A requirements Development Process