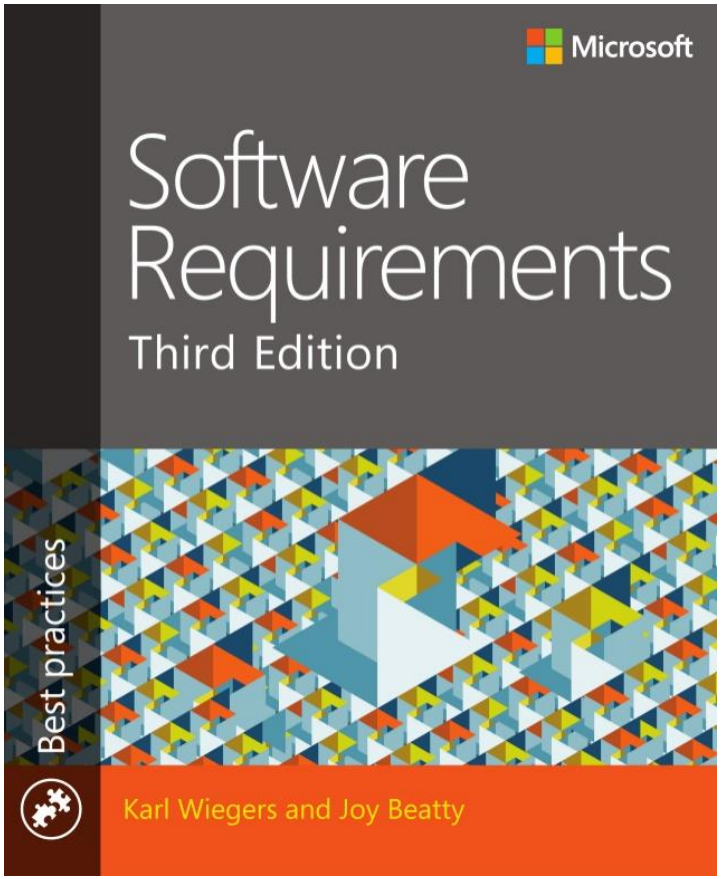
Chapter 1

The essential software requirement



EECS812: Software Requirements Engineering
Professor Hossein Saiedian

This chapter will help you to

- Understand some key terms used in the software requirements domain
- Distinguish *product* requirements from *project* requirements
- Distinguish requirements *development* from requirements *management*
- Be alert to several requirements-related problems that can arise

Favorite definition of requirements

- Sommerville and Sawyer (1997):

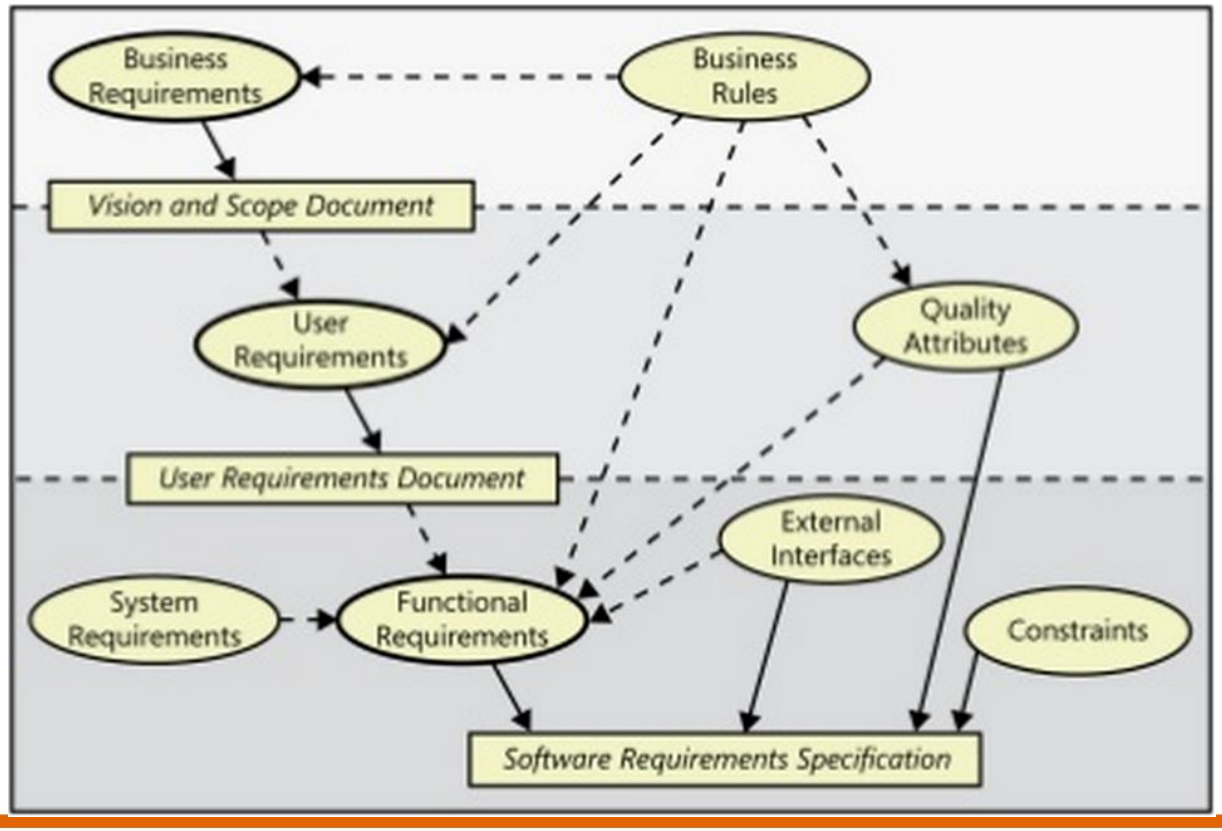
  *Requirements are a specification of **what** should be implemented. They are descriptions of **how** the system should behave, or of a system property or attribute. They may be a constraint on the development process of the system.*

Levels and types of requirements

- Business requirement
- Business rule
- Constraint
- External interface requirement
- Feature
- Functional requirement
- Nonfunctional requirement
- Quality attribute
- System requirement
- User requirement

Relationships among requirements

- Software requirements include three distinct levels



Business requirement

- A high-level business objective of the organization that builds a product, or of a customer who procures it
- Example: An airline wants to reduce airport counter staff costs by 25%

Business rule

- A policy, guideline, standard, or regulation that defines or constrains some aspect of the business
- Not a software requirement in itself, but the origin of several types of software requirements
- Example: Age of majority by postal code

Constraint

- A restriction that is imposed on the choices available to the developer for the design and construction of a product
- Example: The system's business rules should be maintainable without changing code

External interface requirement

- A description of a connection between a software system and a user, another software system, or a hardware device
- Example: A system that uses a third-party product or service to send SMS/push notifications

Feature

- One or more logically related system capabilities that provide value to a user and are described by a set of functional requirements
- Example: Web browser bookmarks, spelling checkers, automatic virus signature updating, etc

Functional requirement

- A description of a behavior that a system will exhibit under specific conditions
- Example: The Passenger shall be able to print boarding passes for all flight segments for which he has checked in

Nonfunctional requirement

- A description of a property or characteristic that a system must exhibit or a constraint that it must respect
- Example: Usability (user-friendliness) of a system

Quality attribute

- A kind of nonfunctional requirement that describes a service or performance characteristic of a product
- Example: A system that is guaranteed to be available 99.99999% of the time

System requirement

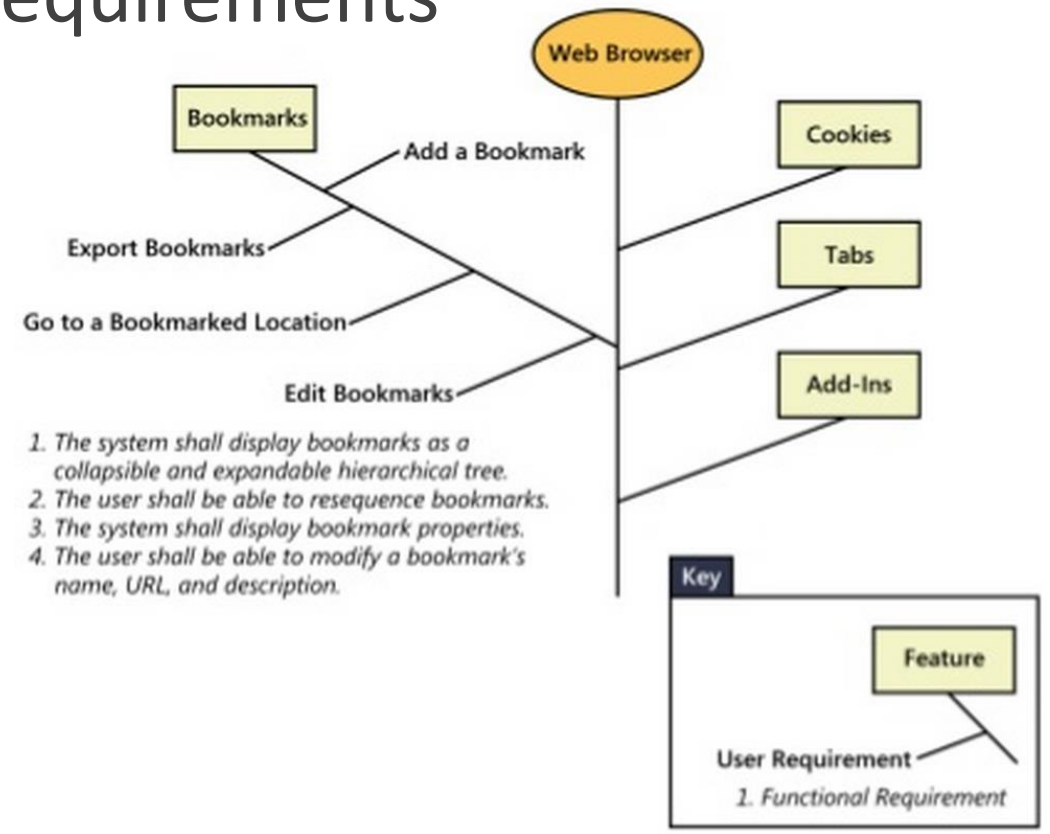
- A top-level requirement for a product that contains multiple subsystems, which could be all software or software and hardware
- Example: Cashier's workstation in a supermarket

User requirement

- A goal or task that specific classes of users must be able to perform with a system, or a desired product attribute
- Example: As a passenger, I want to check in for a flight so I can board my airplane

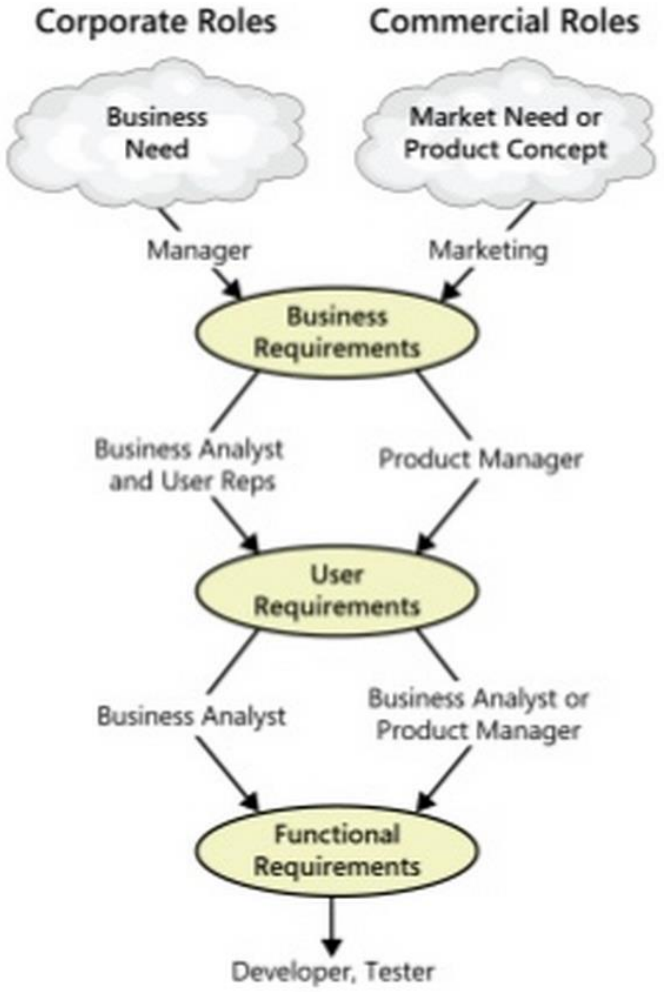
Relationships among features

- Feature decomposition to show user and functional requirements



Working with the three levels

- Example of how different stakeholders participate in requirements development

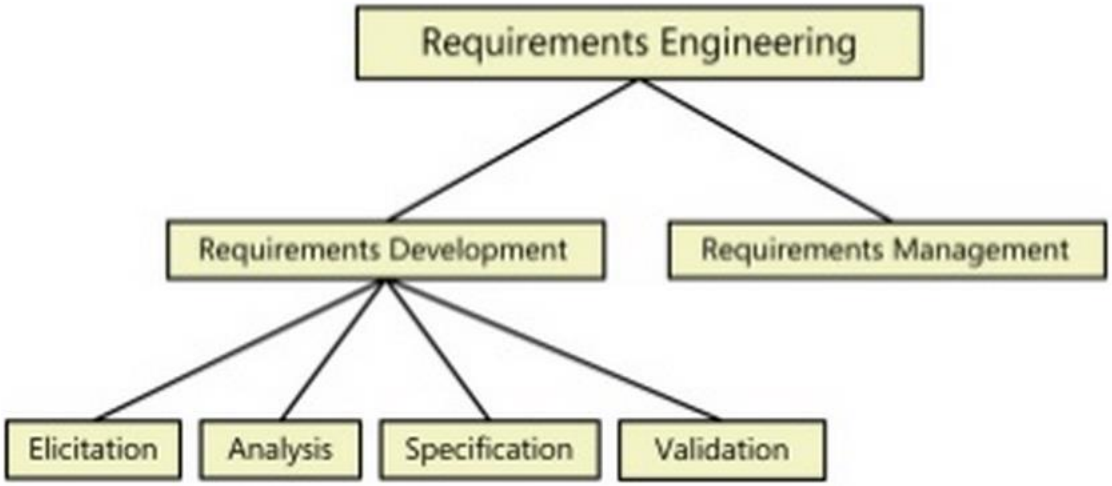


Product vs project requirements

- *Product* requirements describe properties of a software system to be built
- *Project* requirements are:
  - expectations and deliverables
  - *not* a part of the software the team implements
  - necessary to the successful completion of the project as a whole

Requirements development and management

- Subdisciplines of software requirements engineering



Requirements development

- Broken down into *elicitation*, *analysis*, *specification*, and *validation*
- Encompass all the activities involved with exploring, evaluating, documenting, and confirming the requirements for a product

Elicitation

- Encompasses all of the activities involved with discovering requirements like:
  - Interviews
  - Workshops
  - Document analysis
  - Prototyping

Analysis

- Reaching a richer and more precise understanding of each requirement
- Representing sets of requirements in multiple ways

Specification

- Involves representing and storing the collected requirements knowledge in a persistent and well-organized fashion

Validation

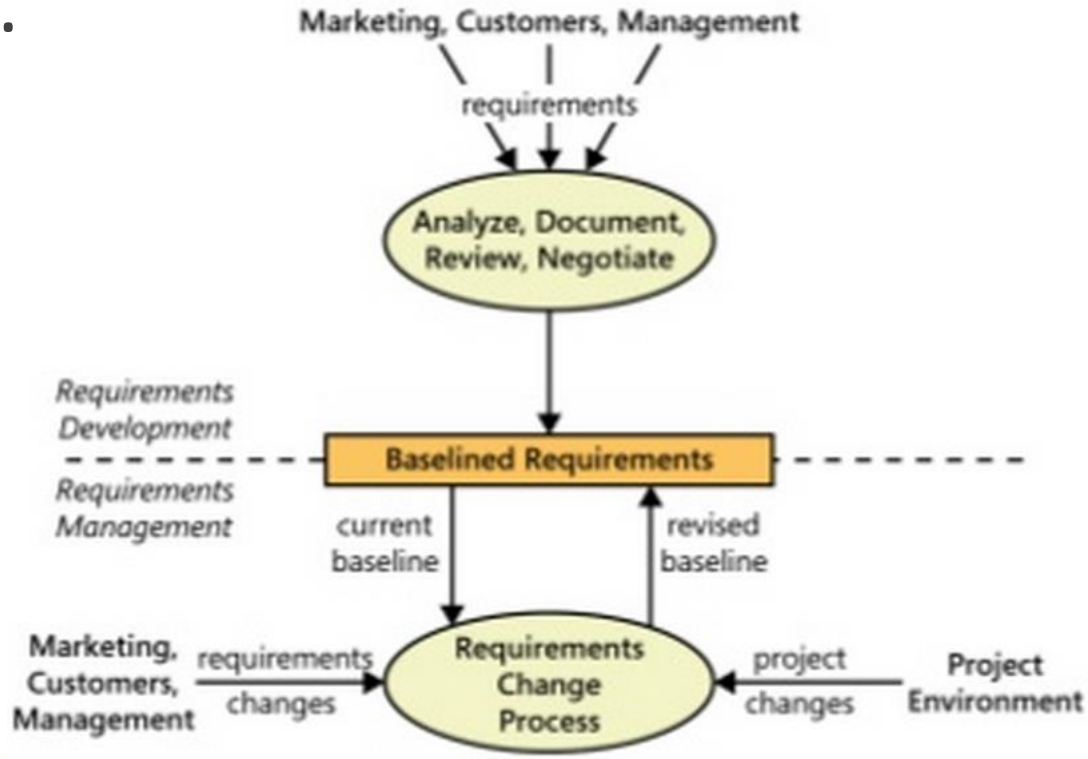
- Confirms that you have the correct set of requirements information that will enable developers to build a solution that satisfies the business objectives

Requirements management

- Defining the requirements baseline for a specific product release or development iteration
- Evaluating the impact of proposed requirements changes and incorporating approved changes into the project in a controlled way
- Keeping project plans current with the requirements as they evolve
- Negotiating new commitments based on the estimated impact of requirements changes
- Defining the relationships and dependencies that exist between requirements
- Tracing individual requirements to their corresponding designs, source code, and tests
- Tracking requirements status and change activity throughout the project

Boundary between development and management

- Another view of the boundary between requirements development and management.



Every project has requirements

- Frederick Brooks (1987):

  The hardest single part of building a software system is deciding precisely what to build. No other part… is as difficult as establishing the detailed technical requirements… so cripples the resulting system if done wrong… is more difficult to rectify later.

Bad requirements

- Insufficient user involvement
- Inaccurate planning
- Creeping requirements
- Ambiguous requirements
- Gold plating
- Overlooked stakeholders

Benefits from a high quality process

- Fewer defects in requirements and in the delivered product
- Reduced development rework
- Faster development and delivery
- Fewer unnecessary and unused features
- Lower enhancement costs
- Fewer miscommunications
- Reduced scope creep
- Reduced project chaos
- Higher customer and team member satisfaction
- Products that do what they're supposed to do

Summary

- Quality requirements are key to a quality software product
- Costs of a software project can be substantially reduced by spending the time to produce accurate requirements
- Requirements must be documented