FPT UNIVERSITY

ĐẠI HỌC FPT CẦN THƠ

CHAPTER 1
The essential of software requirement

Objectives

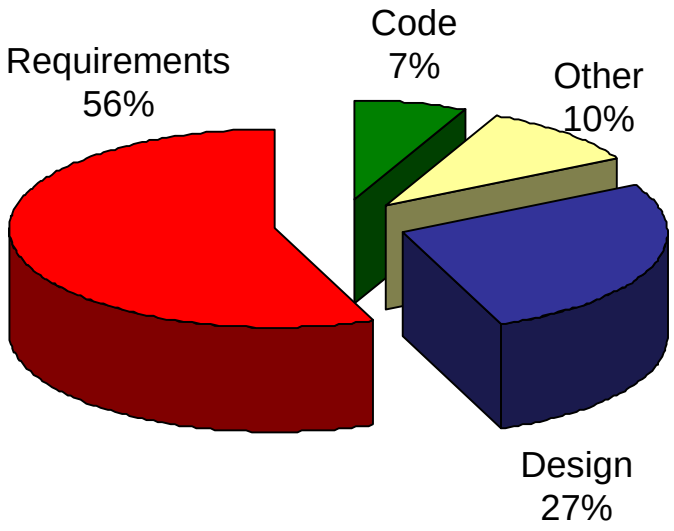
- Understand some key terms used in the software requirements domain.
- Distinguish product requirements from project requirements.
- Distinguish requirements development from requirements management.
- Be alert to several requirements-related problems that can arise

Contents

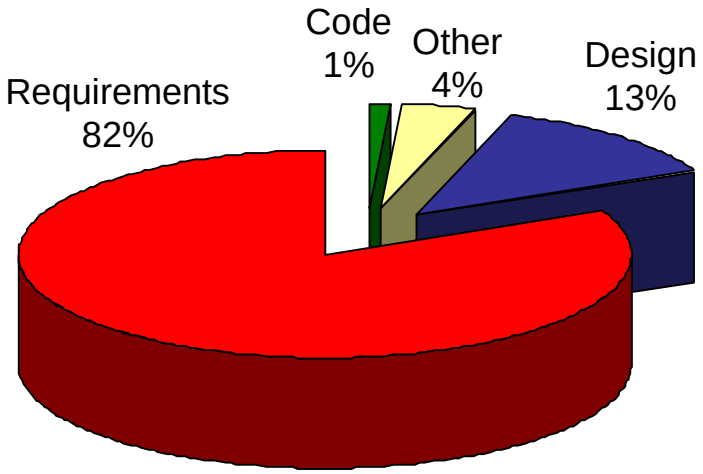
1. Why is requirement engineering?
2. Software requirements defined.
3. Requirements development and management.
4. When bad requirements happen to good people.
5. Benefits from a high-quality requirements process.

Why is requirement engineering?

- **Distribution of Defects**



- **Distribution of Effort to Fix Defects**



Source: Martin & Leffinwell

Software requirements defined

- The errors introduced during requirements activities account for 40 to 50% of all defects found in software product.

- Inadequate user input and shortcomings in specifying and managing customer requirements are major contributors to successful projects.

Software requirements defined

- Brian Lawrence: “anything that drives design choices” (*Lawrence 1997*)

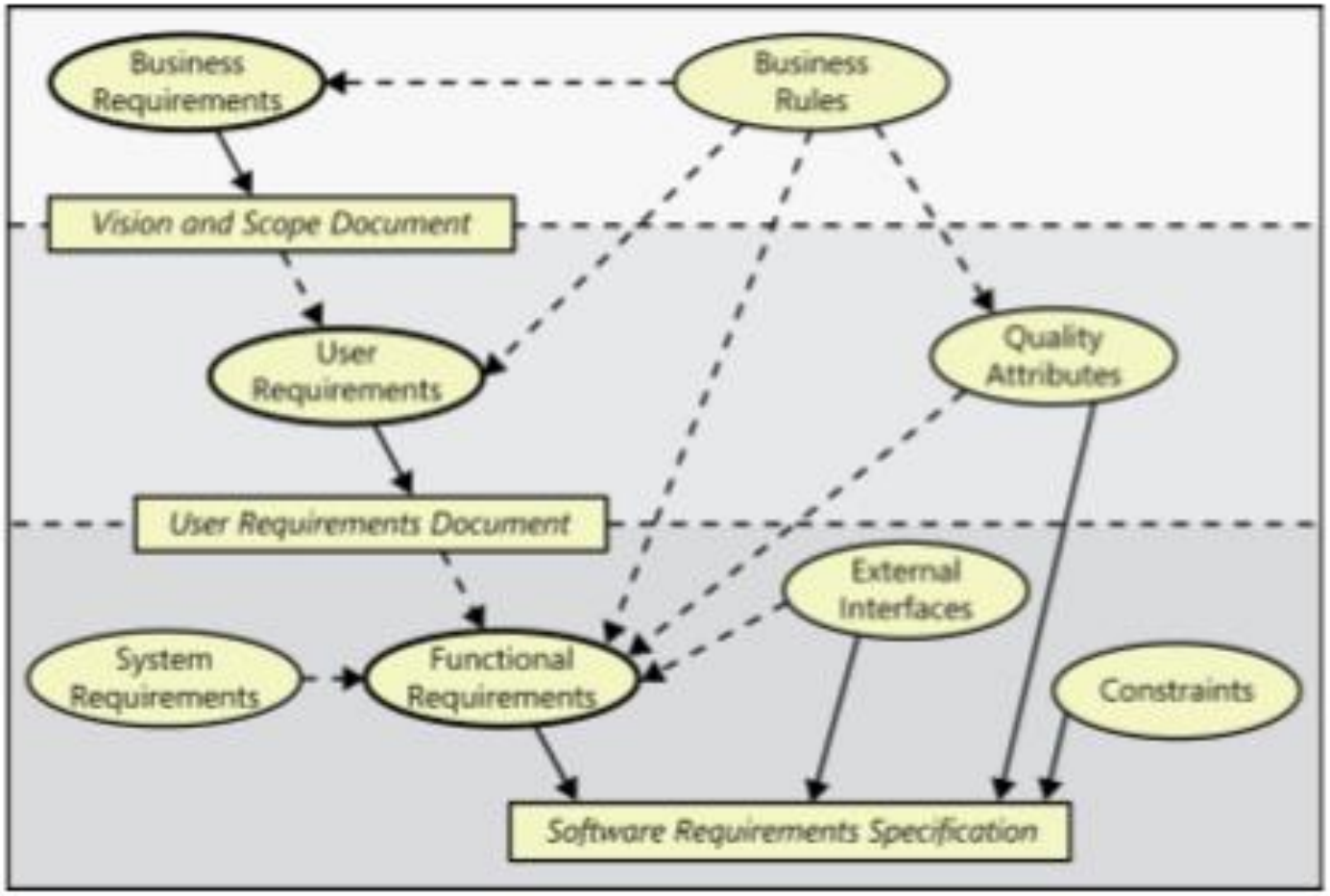
- “**Requirements are a specification of what should be implemented. They are descriptions of how the system should behave, or of a system property or attribute. They may be a constraint on the development process of the system**” *Ian Sommerville and Pete Sawyer (1997)*

Types of requirements

TABLE 1-1 Some types of requirements information

Term	Definition
Business requirement	A high-level business objective of the organization that builds a product or of a customer who procures it.
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.
Constraint	A restriction that is imposed on the choices available to the developer for the design and construction of a product.
External interface requirement	A description of a connection between a software system and a user, another software system, or a hardware device.
Feature	One or more logically related system capabilities that provide value to a user and are described by a set of functional requirements.
Functional requirement	A description of a behavior that a system will exhibit under specific conditions.
Nonfunctional requirement	A description of a property or characteristic that a system must exhibit or a constraint that it must respect.
Quality attribute	A kind of nonfunctional requirement that describes a service or performance characteristic of a product.
System requirement	A top-level requirement for a product that contains multiple subsystems, which could be all software or software and hardware.
User requirement	A goal or task that specific classes of users must be able to perform with a system, or a desired product attribute.

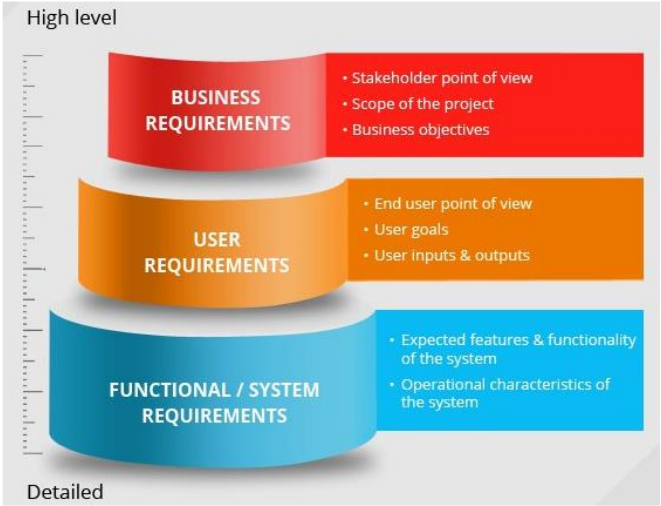
Levels and types of requirements



Relationships among several types of requirements information. Solid arrows mean “are stored in”; dotted arrows mean “are the origin of” or “influence”.

Business requirements

- Biz requirements describe why the organization is implementing the system – the biz benefits the organization hopes to achieve.
- *Suppose an airline wants to reduce airport counter staff costs by 25%. This goal might lead to the idea of building a kiosk that passengers can use check in their flights at the airport.*
- Biz requirements typically come from the funding sponsor for a project, the acquiring customers, the manager of the actual users, the marketing department, or a product visionary.



We will study more details in chapter 5.

User requirements

- User requirements describe goals or tasks the users must be able to perform with the product that will provide value to someone.
- The domain of user requirements includes descriptions of product attributes or characteristics that are important to user satisfaction.
- User requirements describe *what* the user will be able to do with the system.
- Ways to represent user requirements includes *user cases*, *user stories*, or *event-response tables*.

We will study more details in chapter 8.

Functional requirements

- Functional requirements specify the behaviors the product will exhibit under specific conditions.
- They describe *what* the developers must implement to enable users to accomplish their tasks (user requirements), thereby satisfying the biz requirements.
- Functional requirements often are written in the form of the traditional "shall" statements: *"The passenger shall be able to print boarding passes for all flight segments for which he has checked in"*.

Functional Requirement No.	Function Requirement Description
FR 1	User should be able to enter Sales Data
FR 2	Sales Reports should be generated every 24 hours
FR 3	API interface to Invoice System

Biz analyst (BA)

- The BA documents functional requirements in a *SW requirements specification* (SRS), which describes as fully as necessary the expected behavior of the SW system.
- The SRS is used in development, testing, quality assurance, project management, and related project functions.

We will study more details in chapter 10.

System requirements

- System requirements describe the requirements for a product that is composed of multiple components or subsystems.
- A system can be all SW or it can include both SW and HW subsystems.
- People and processes are part of a system, too, so certain system functions might be allocated to human being.

REQUIREMENTS	OS	CPU	RAM	HDD
Minimum System Requirements	Windows 7 SP1 Windows 8 Windows 8.1 Windows 10	Dual Core 1.6GHz	1GB	1.5GB
Recommended System Requirements	Windows 7 SP1 Windows 8 Windows 8.1 Windows 10	Intel Core 2 Duo (2GHz) or equivalent processor	2GB	2GB
Software requirements	Internet Explorer 10 and higher			
Integrates with	Google Chrome version 34 and higher Mozilla Firefox version 30 and higher Skype version 6.3 and higher Thunderbird version 14 and higher Outlook 2007, 2010, 2013, 2016			

Business rules

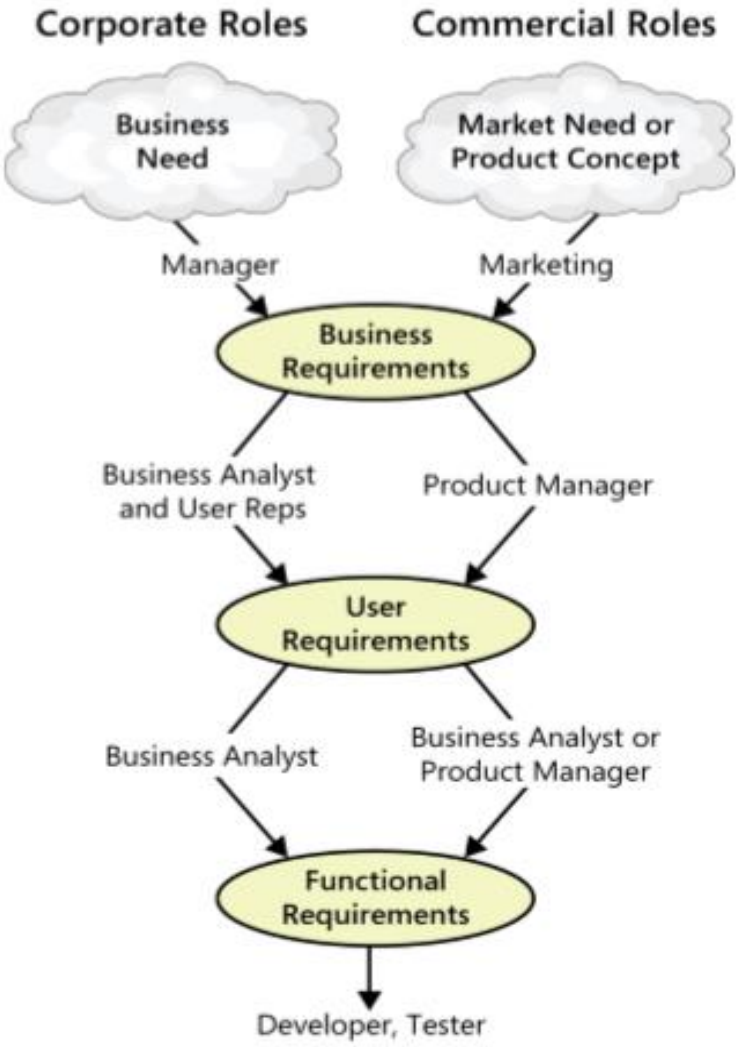
- Business rules include corporate policies, government regulations, industry standards, and computational algorithms.
- Biz rules are not themselves SW requirements because they have an existence beyond the boudaries of any specific SW application.

Business Rules
and Process Rules

Feature

- A feature consists of one or more logically related system capabilities that provide value to a user and are described by a set of functional requirements.
- Web browser bookmarks, spelling checkers, … are examples of features.

The participation of stakeholders in requirements development



- This figure illustrates how various stakeholders might participate in eliciting the 3 levels of requirements..

Product vs. project requirements

- Product requirements:
  - properties of a software system to be built
- Projects requirements:
  - do have other expectations and deliverables that are not a part of the software the team implements, but that are necessary to the successful completion of the project as a whole.

Project requirements

- Physical resources the development team needs, such as workstations, HW devices, testing labs, testing tools and equipment, team room, ….
- Staff training needs.
- User documentation, including training materials, tutorials, reference manuals, and release notes.
- Support documentation, such as help desk resources and field maintenance and service information for HW devices.
- Requirements and procedures for releasing the product, installing it in the operating environment, configuring it, and testing the installation.
- ….

Requirements development and management

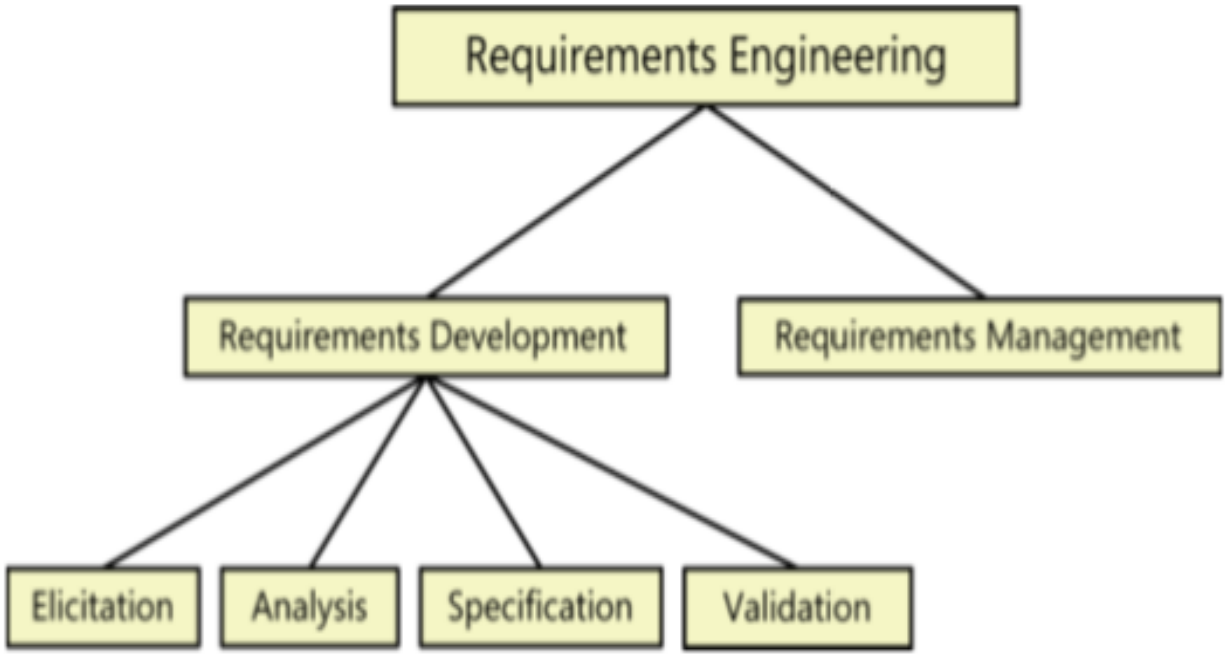


FIGURE 1-4 Subdisciplines of software requirements engineering.

- Depending on the life cycle, you will perform these activities at different times in the project and to varying of depth or detail.

Requirements development

- We subdivide the requirements development into *elicitation, analysis, specification,* and *validation.*
- ***Elicitation*** encompasses all of the activities involved with discovering requirements, such as interviews, workshops, document analysis, prototyping, and others. The key actions are:
  - Identifying the product's expected user classes and other stakeholders.
  - Understanding user tasks and goals and the biz objectives with which those tasks align.
  - Learning about environment in which the new product will be used.
  - Working with individuals who represent each user class to understand their functionality needs and their quality expectations.

Usage-centric or product-centric?

- Requirements elicitation typically takes either a usage-centric or product-centric approach.
- The usage-centric focuses on understanding and exploring user goals to derive the necessary system functionality.
- The product-centric focuses on defining features that you expect will lead to marketplace or biz success.
- A risk with product-centric is that you might implement features that don't get used much, even if they seemed like a good idea at the time.
- We recommend understanding biz objectives and users goals first, then using that insight to determine the appropriate product feature and characteristics.

Requirements development

- ***Analysis***: Analyzing requirements involved reaching a richer and more precise understanding of each requirement and representing set of requirements in multiple ways.
- Following are the principal activities:
  - Analyzing the information received from users to distinguish their task goals from functional requirements, quality expectations, biz rules, suggestion solutions, and other information.
  - Decomposing high-level requirements into an appropriate level of detail.
  - Deriving functional requirements from other requirements.
  - Understanding the relative importance of quality attributes.
  - Allocating requirements to SW components defined in the system architecture.
  - Negotiating implements priorities.
  - Identifying gaps in requirements or unnecessary requirements.

Requirements development

- ***Specification***: Requirements spec involves representing and storing the collected requirements knowledge in a persistent and well-organized fashion. The principal activity is:
  - Translating the collected user needs into written requirements and diagrams suitable for comprehension, review, and use by their intended audiences.
- ***Validation***: Requirements validation confirms that you have the correct set of requirements information that will enable developers to build a solution. The central activities are:
  - Reviewing the documented requirements to correct any problems before the development group accept them.
  - Developing acceptable tests and criteria to confirm that a product based on the requirements would meet customer needs and achieve the biz objectives.

Requirements management

Requirements management activities include:

- Defining the requirements baseline, a snapshot in time that represents an agreed-upon, reviewed, and approved set of functional and nonfunctional requirements, often 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 test.
- Tracking requirements status and change activity throughout the project.

Requirements management

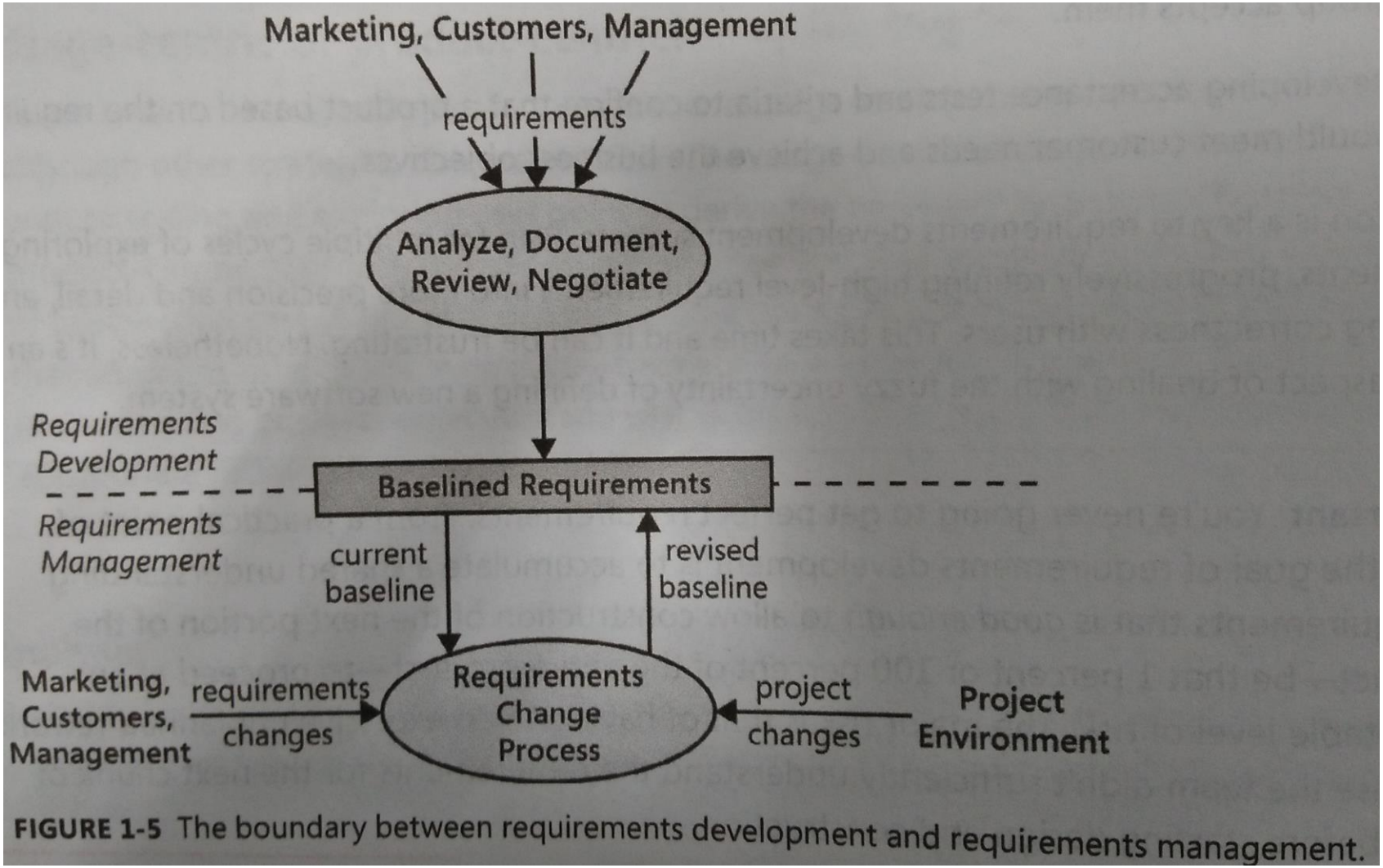


FIGURE 1-5 The boundary between requirements development and requirements management.

Every project has requirements

- *The hardest single part of building a software system is deciding precisely what to build. No other part of conceptual work is as difficult as establishing the detailed technical requirements, including all the interfaces to people, to machines, and to other SW systems. No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later.*
- Every SW-containing system has stakeholders who rely on it. The time spent understanding their needs is a high-leverage investment in project success. If a project team does not have written representations of requirements that the stakeholders agree to, how can developers be sure to satisfy those stakeholders?

When bad requirements happen to good people

- The major consequence of requirements problem is rework – doing again something that you thought was already done – late in development or after release.
- Rework often consumes 30-50% of total development cost, and requirements errors can account for 70-80% of the rework cost.
- The most common requirements risks are described below:
  - Insufficient user involvement
  - Inaccurate planning
  - Creeping user requirements
  - Ambiguous requirements
  - Gold plating
  - Overlooked stakeholders

When bad requirements happen to good people

- Insufficient user involvement:
  - Customers often don't understand why it is so essential to work hard on eliciting requirements and assuring their quality.
  - Developers might not emphasize user involvement, because they think they already understand what the users need.
  - Customers are busy so they don't have enough time to involve in requirements development.
- Inaccurate planning:
  - Vague, poorly understood requirements lead to overly optimistic estimates, which come back to haunt you when the inevitable overruns occur.
  - The top contributors to poor software cost estimation are frequent changes, missing requirements, insufficient communication with users, poor specification of requirements and insufficient requirements analysis (Davis 1995).

When bad requirements happen to good people

- Creeping user requirements
  - As requirements evolve during development, projects often exeed their planned schedules and budgets (which are nearly always too optimistic anyway). To manage scope creep, begin with a clear statement of the project's biz objectives, strategic vision, scope, limitations, and success criteria.
  - Requirements will change and grow. The PM should build contingency buffers into schedules so the first new requirement that comes along doesn't derail the schedule (Wiegers 2007).
  - Agile projects take the approach of adjusting the scope for a certain iteration.

When bad requirements happen to good people

- Ambiguous requirements:
  - One symptom of ambiguity in requirements is that a reader can interpret a requirement statement in serveral ways (Lawrence 1996).
  - Ambiguity leads to different expectations on the part of various stakeholders. Some of them are then surprised at whatever is delivered. Ambiguous requirements cause wasted time when developers implement a solution for the wrong problem.
  - One way to ferret out ambiguity is to have people who represent different perspectives inspect the requirements (Wieger 2002).

When bad requirements happen to good people

- Gold plating
  - *Gold plating* takes place when a developer adds functionality that wasn't in the requirements specification but which developer believes "the users are just going to love." If users don't care about this functionality, the time spent implementing it is wasted. *(need to be approved by stakeholders).*
  - Customers sometimes request certain features that look attractive but add little value to product. Everything you build costs time and money, so you need to maximize the delivered value.
- Overlooked stakeholders
  - Most project have several groups of users who might use different subset of features, have different frequencies of use, or have varying levels of experience. If you don't identify the important user classes for your product early on, some user needs won't be met.
  - You need to think about support different users with different functionalities.

Benefits from a high-quality requirements process

- Sound requirements processes emphasize a collaborative approach to product development that involves stakeholders in a partnership throughout the project.
- Customer involvement reduces the expectation gap between what the customer really needs and what the developer delivers.
- The cost of better requirements includes developing new procedures and document templates, training the team, and buying tools.
- Your greatest investment is the time to spend on requirements engineering tasks.

Benefits from a high-quality requirements process

The potential payoff includes:

- 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.

THE END
THANK YOU!