COURSE NAME

SOFTWARE REQUIREMENT ENGINEERING

CSC 4126

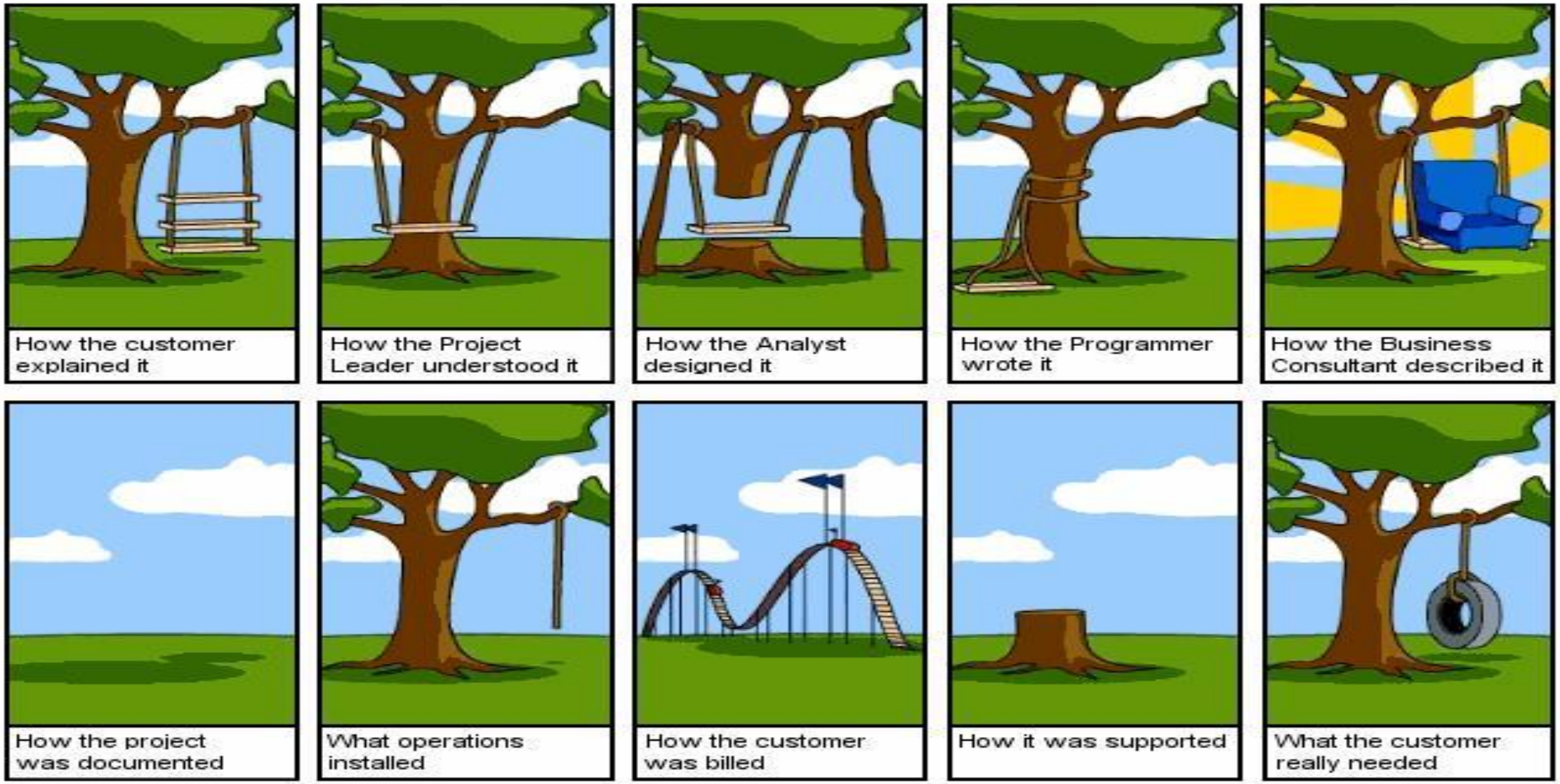
(UNDERGRADUATE)

CHAPTER 1

THE ESSENTIAL OF SOFTWARE REQUIREMENTS

DR. MOHAMMAD MAHMUDUL HASAN
ASSOCIATE PROFESSOR, CS, FST, AIUB
Web: http://cs.aiub.edu/profile/m.hasan

THE CURRENT COMMON PROBLEMS



THE CURRENT COMMON PROBLEMS

- Business objective: The project's business objectives, vision, and scope were never clearly defined.
- Lack of communication: customers were too busy to spend time working with analysts or developers on the requirements (what users needed to accomplish with the software)
- Requirements Prioritization: Customers claimed that all requirements were critical, so they didn't prioritize them.
- Ambiguous and Incompleteness: Developers encountered ambiguities and missing information when coding, so they had to guess
- Customer approval: Your customers never approved the requirements.

THE CURRENT COMMON PROBLEMS (CNTD.)

- Change requirements: Your customers approved the requirements for a release or iteration and then changed them continually.
- The project scope increased: as requirements changes were accepted, but the schedule slipped because no additional resources were provided and no functionality was removed.
- Requested requirements changes got lost: no one knew the status of a particular change request.
- Functionality use: Customers requested certain functionality and developers built it, but no one ever uses it.

WHAT IS "REQUIREMENT"?

- A requirement is a property that a product must have to provide value to a stakeholder
- Requirements are a specification of what should be implemented. They are descriptions of a system property or attribute or how the system should behave. They may be a constraint on the development process of the system.
- Requirements encompass
  - the user's view of the external system behaviour
  - the developer's view of some internal characteristics
- Software requirements include a time dimension
  - They could be present tense, describing the current system's capabilities
  - Near-term (high priority) or hypothetical (low priority) future
  - They could even be past tense, referring to needs that were once specified and then discarded

BUSINESS REQUIREMENTS

- Business requirements describe why the organization is implementing the system — the business benefits the organization hopes to achieve
- The focus is on the business objectives of the organization or the customer who requests the system (business requirements collected from multiple sources might conflict)
- *Suppose an airline wants to reduce airport counter staff costs by 25 percent. This goal might lead to the idea of building a kiosk that passengers can use to check in for their flights at the airport.*
- Self-service technologies (SSTs) have been applied to many areas of business
- Business requirements typically come from the funding sponsor for a project, the acquiring customer, the manager of the actual users, the marketing department (to promote a product or service), or a product visionary

USER REQUIREMENTS

- User requirements describe goals or tasks the users must be able to perform with the product that will provide value to someone
  - Includes descriptions of product attributes or characteristics that are important to user satisfaction
  - Represent user requirements include use cases and user stories
  - User representatives will provide this information
  - Describes what the user will be able to do with the system
  - *An example of a use case is "Check in for a flight" using an airline's website or a kiosk at the airport. Written as a user story, the same user requirement might read: "**As a passenger, I want to check in for a flight so I can board my airplane."***
  - *Other users are airline employees, FDA, etc.*

FUNCTIONAL REQUIREMENTS

- Functional requirements specify the behaviours 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 business 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" or "If the Passenger's profile does not indicate a seating preference, the reservation system shall assign a seat."

  - A restriction that is imposed on the choices available to the developer for the design and construction of a product, is a CONSTRAINT

"The system shall be developed using open source tools and shall run on Linux OS"

LEVELS OF REQUIREMENTS

- Software requirements include three distinct levels:
  - Business requirements
  - User requirements
  - Functional requirements

- This alignment among the three levels of requirements is essential for project success

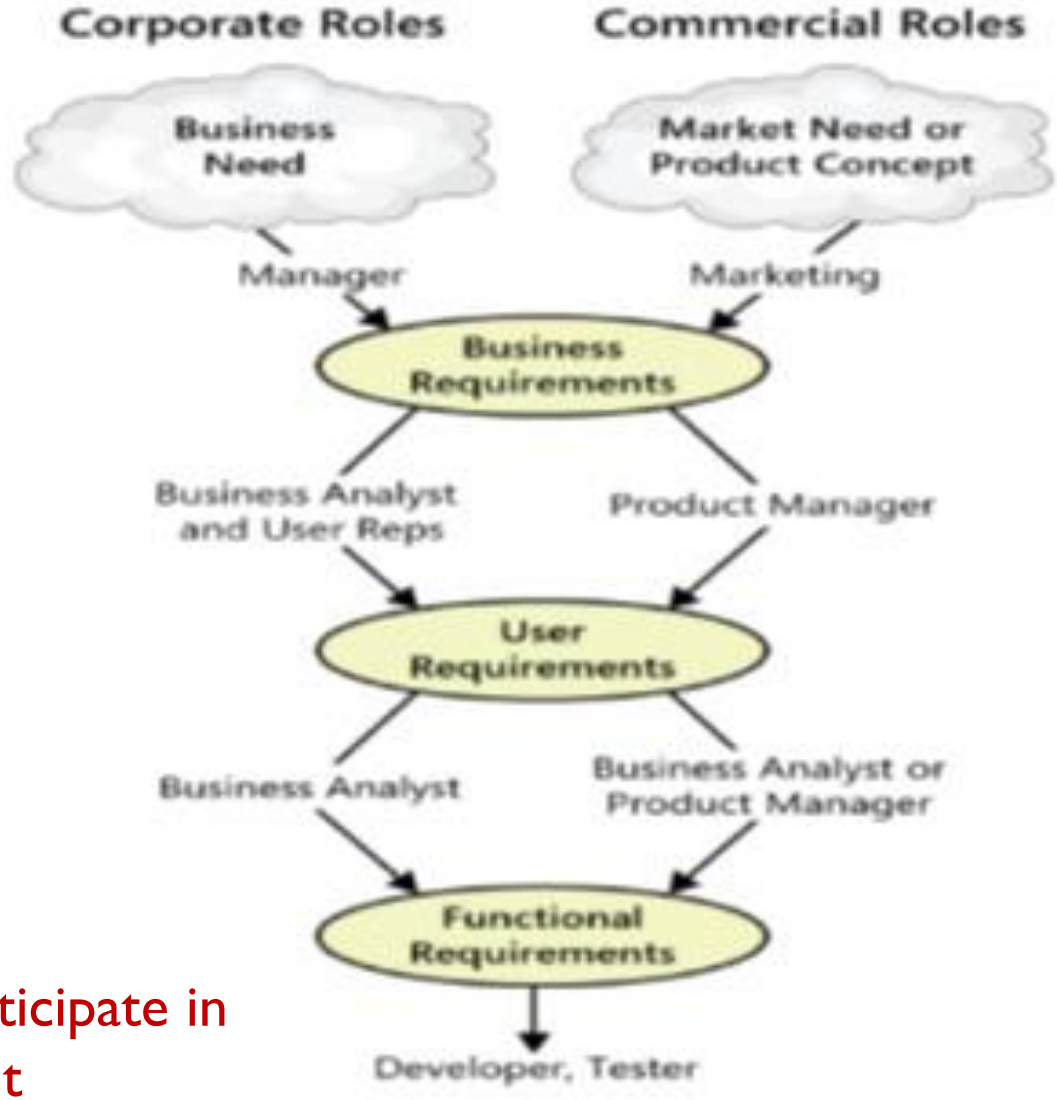


Figure: Different stakeholders participate in requirements development

SYSTEM REQUIREMENTS

- System requirements describe the requirements for a product that is composed of multiple components or subsystems.
- A system can be all software or it can include both software and hardware subsystems (e.g. biometric)
- People and processes are part of a system, so certain system functions might be allocated to the human being
- *A good example of a "system" is the cashier's workstation in a supermarket. There's a bar code scanner and card reader. Also the cashier has a keyboard, a display, and a cash drawer, and printer to print to purchase receipt*
- The requirements for the system or product as a whole, then, lead the business analyst to derive specific functionality that must be allocated to one or another of those component subsystems, as well as demanding an understanding of the interfaces between them

NON-FUNCTIONAL REQUIREMENTS

- Non-functional requirement are also known as quality attributes, product requirements that describes a service or performance characteristic of a product ("– ity, ilities.")
- Describe the product's characteristics in various dimensions that are important either to users or to developers and maintainers, such as performance, safety, availability, and portability (chances of conflicts within non-functional requirements are fairly high)
- Other classes of non-functional requirements describe external interfaces between the system and the outside world (usefulness, flexibility, reliability)
- These include connections to other software systems, hardware components, and users, as well as communication interfaces (interoperability)
- Design and implementation constraints impose restrictions on the options available to the developer during construction of the product (security)

BUSINESS RULES

- Business rules include corporate policies, government regulations, industry standards, and computational algorithms
- Business rules are not themselves software requirements because they have an existence beyond the boundaries of any specific software application
- They often dictate that the system must contain functionality to comply with the relevant rules (e.g. online purchase limit)
- A restriction that is imposed on the choices available to the developer for the design and construction of a product (e.g. no red colour in webpage – Inverse requirements)
- Sometimes, as with corporate security policies, business rules are the origin of specific quality attributes that are then implemented in functionality (e.g. firewall is functional requirement and security is quality requirements)
- Therefore, you can trace the origin of certain functional requirements back to a particular business rule

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
- A feature can encompass multiple user requirements, each of which implies that certain functional requirements must be implemented to allow the user to perform the task described by each user requirement

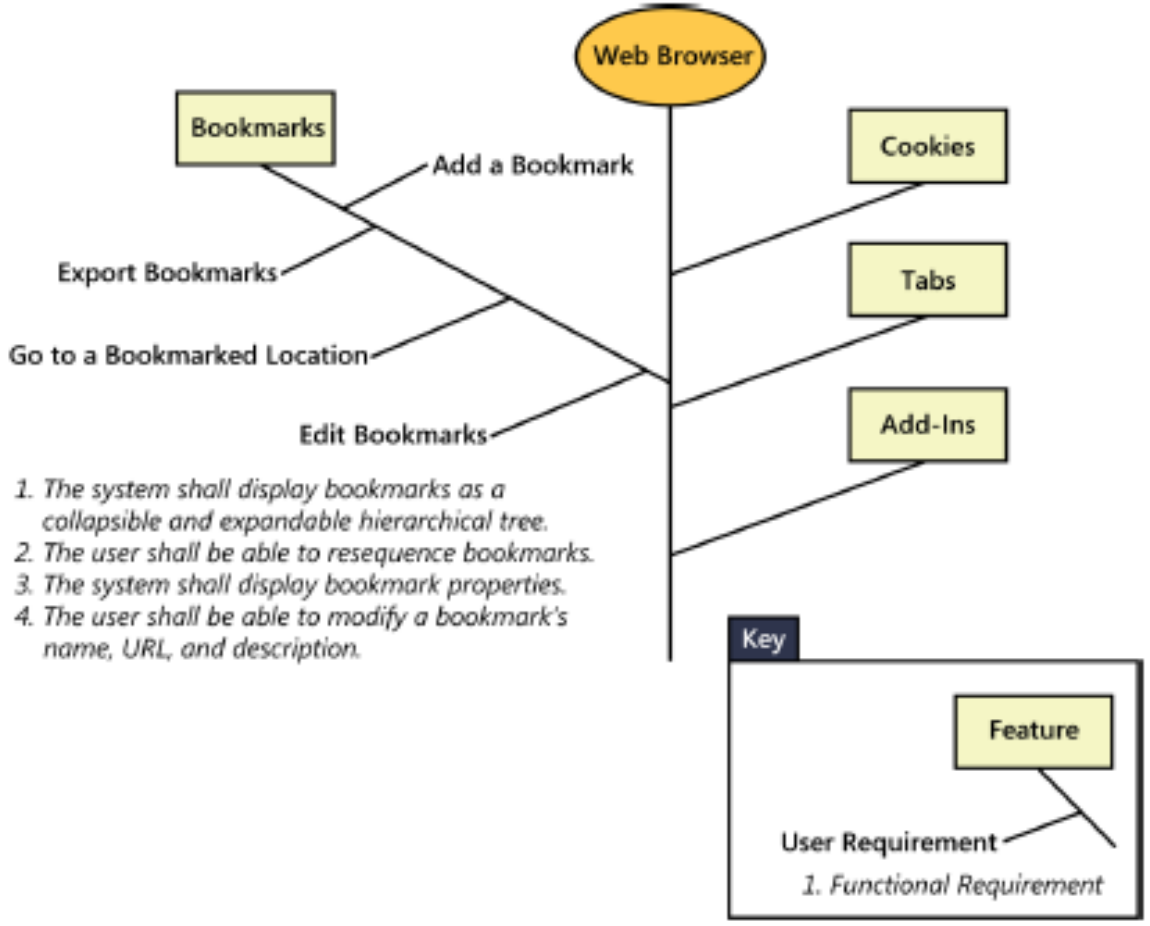


FIGURE 1-2 Relationships among features, user requirements, and functional requirements.

RELATIONSHIPS

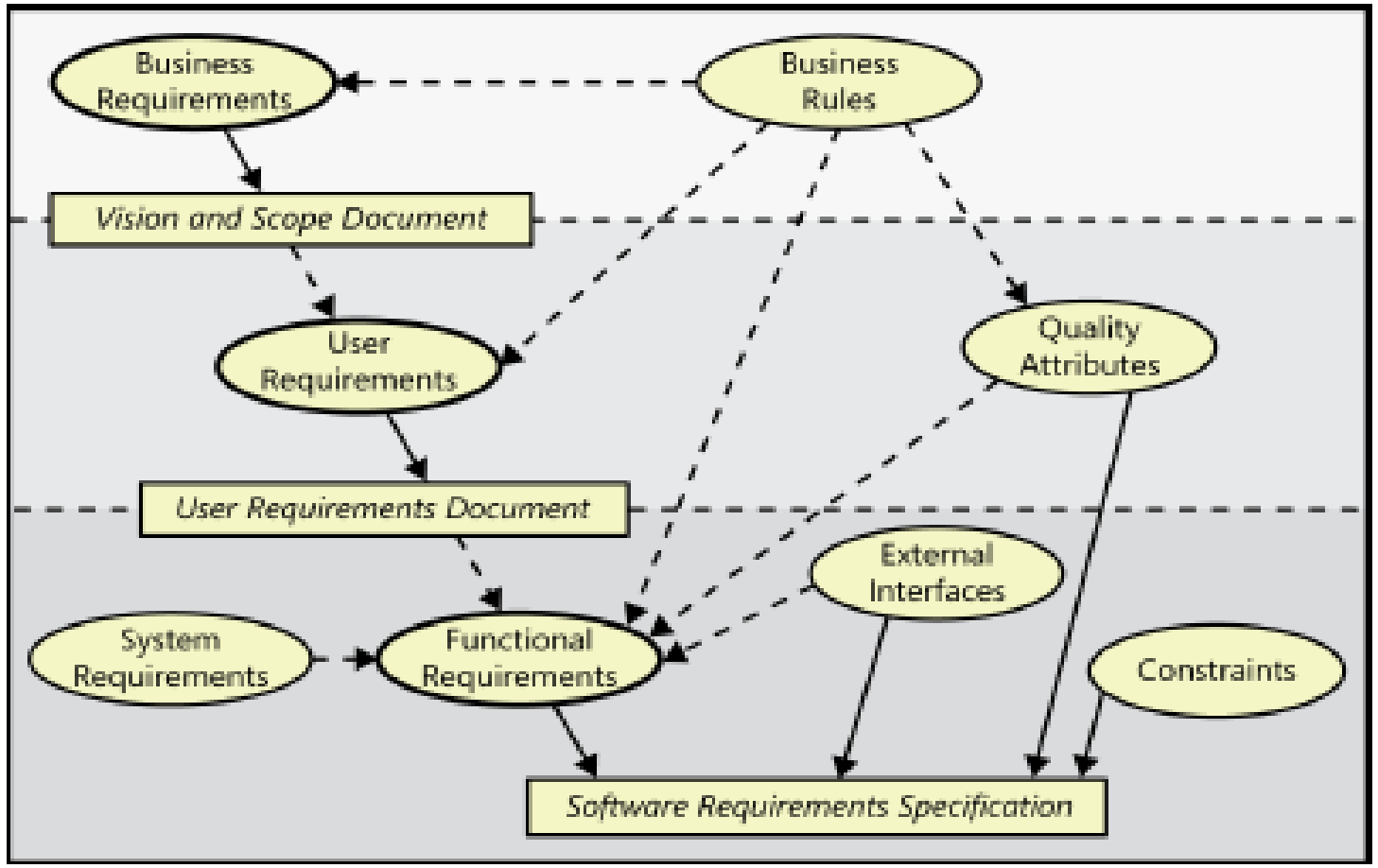


FIGURE 1-1 Relationships among several types of requirements information. Solid arrows mean "are stored in";

PRODUCT VS. PROJECT REQUIREMENTS

- Requirements that describe properties of a software system to be built are called product requirements.
- Projects certainly 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. These are project requirements but not product requirements. (maintain schedule deadline of task)
- An SRS houses the product requirements, but it should not include design or implementation details (other than known constraints), project plans, test plans, or similar information.
- Separate out such items so that requirements development activities can focus on understanding what the team intends to build.

PROJECT REQUIREMENTS

- Physical resources the development team needs, such as workstations, special hardware devices, testing labs, testing tools and equipment, team rooms, and videoconferencing equipment.
- Staff training (User documentation, including training materials, tutorials, reference manuals, and release notes)
- Infrastructure changes needed in the operating environment.
- Requirements and procedures for releasing the product, installing it in the operating environment, configuring it, and testing the installation.
- Beta testing, manufacturing, packaging, marketing, and distribution requirements.
- Customer service-level agreements (SLAs).
- Requirements for obtaining legal protection (patents, trademarks, or copyrights) for intellectual property related to the software.

REQUIREMENTS ENGINEERING PHASES AND ACTIVITIES

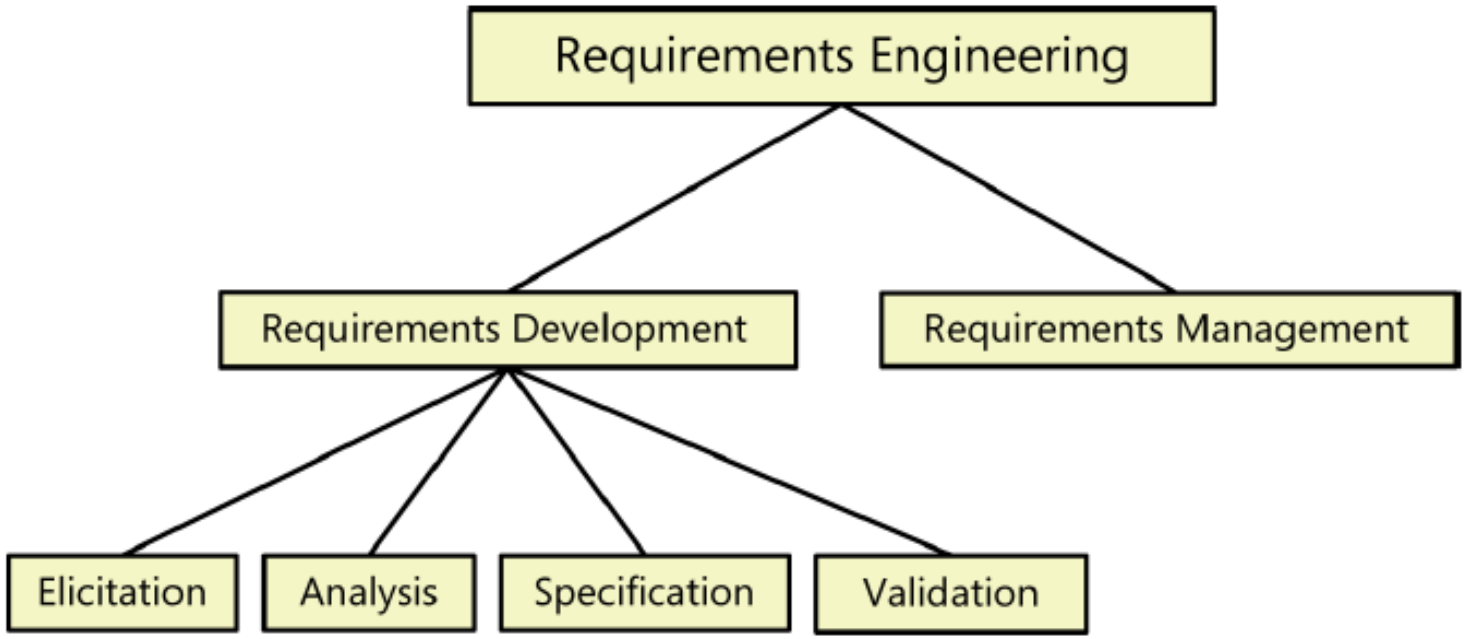
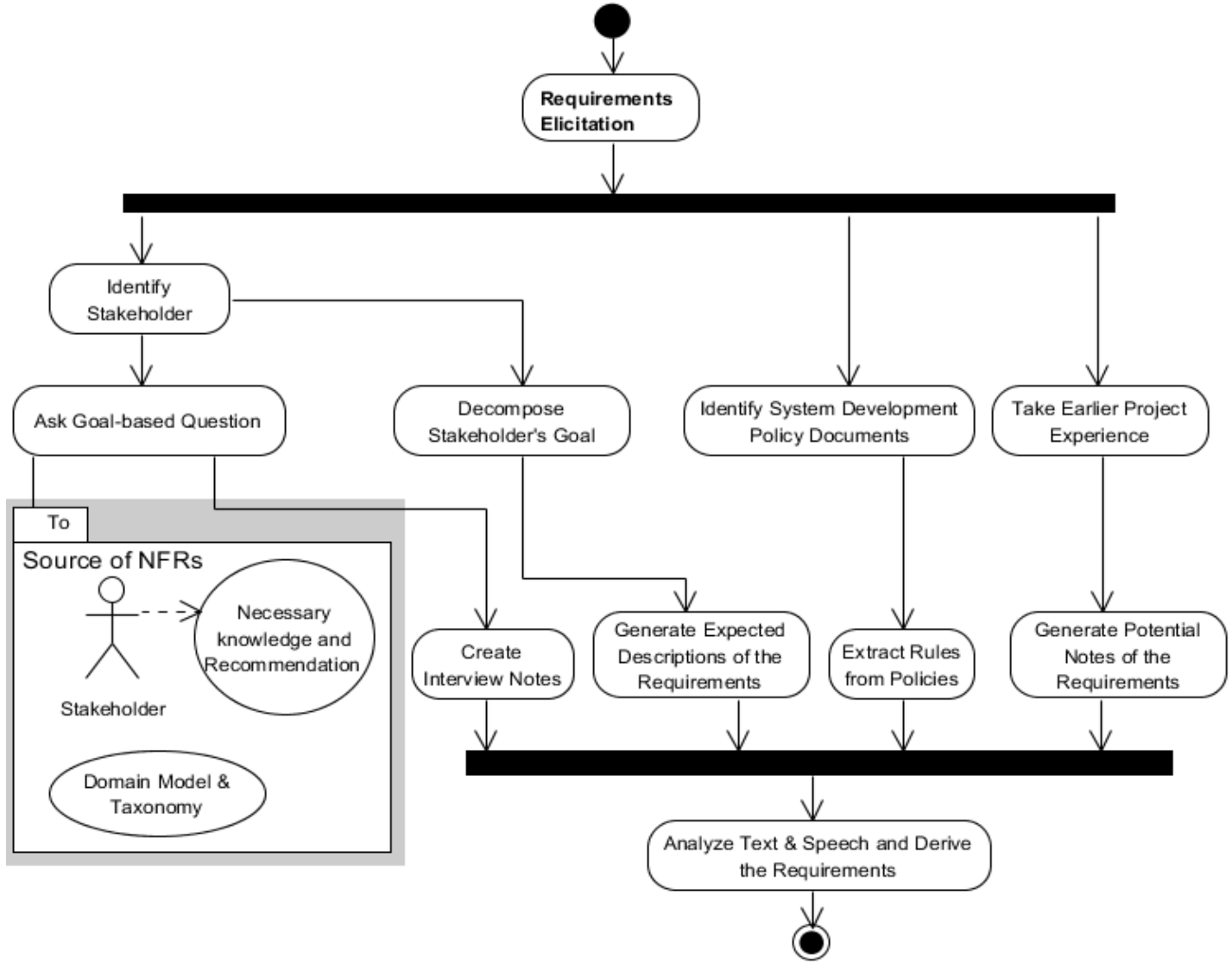


FIGURE 1-4 Subdisciplines of software requirements engineering.

REQUIREMENTS DEVELOPMENT: **ELICITATION**

- Identifying the product's expected user classes and other stakeholders.
- Understanding user tasks and goals and the business objectives with which those tasks align.
- Learning about the 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.



REQUIREMENTS DEVELOPMENT: **ANALYSIS**

- Model the application environment
- Analysing the information received from users to distinguish their task goals into functional requirements, quality expectations, business rules, suggested solutions, etc.
- Decomposing high-level requirements into an appropriate level of detail
- Allocating requirements to software components defined in the system architecture
- Negotiating requirements priority and their implementation priorities

REQUIREMENTS DEVELOPMENT: **SPECIFICATION**

- The business analyst documents requirements in a software requirements specification (SRS).
  - Transcribing the collected user needs into **written requirements** and **diagrams** suitable for comprehension, review, and use by their intended audiences.
  - The SRS describes as fully as necessary the expected behaviour of the software system.
  - The SRS is used in development, testing, quality assurance, project management, and related project functions.
  - Adopt requirement document templates (well defined standard of writing requirements)
  - Identify requirement origins
  - Uniquely label each requirement and cross-cutting requirements concern

REQUIREMENTS DEVELOPMENT: **VALIDATION**

- Reviewing the documented requirements (SRS) to correct any problems before the development group accepts them (feasible, consistent, complete)
- Consistency means that no requirements should be contradictory; Completeness means that no services or constraints which are needed have been missed out. Also, discard unnecessary requirements
- Developing acceptance tests and criteria to confirm that a product based on the requirements would meet customer needs and achieve the business objectives.
- Simulate the requirements (e.g. wireframing) to find errors.

A REQUIREMENTS DEVELOPMENT PROCESS FRAMEWORK

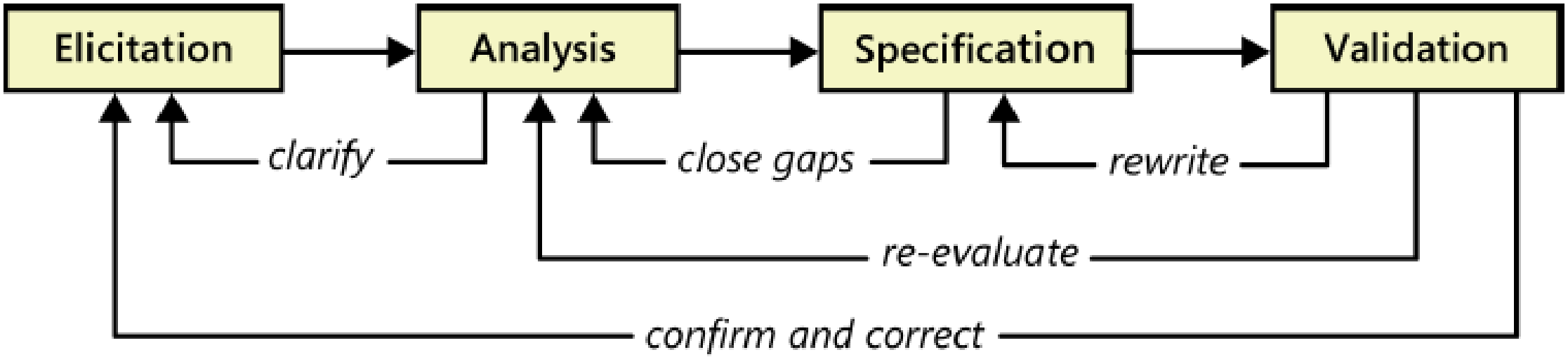


FIGURE 3-1 Requirements development is an iterative process.

A REQUIREMENTS DEVELOPMENT PROCESS FRAMEWORK

1. Define business requirements
2. Identify user classes
3. Identify user representatives
4. Identify requirements decision makers
5. Plan elicitation
6. Identify user requirements
7. Prioritize user requirements

→

8. Flesh out user requirements
9. Derive functional requirements
10. Model the requirements
11. Specify nonfunctional requirements
12. Review requirements
13. Develop prototypes
14. Develop or evolve architecture
15. Allocate requirements to components
16. Develop tests from requirements
17. Validate user requirements, functional requirements, nonfunctional requirements, analysis models, and prototypes

Repeat for iteration 2

Repeat for iteration 3

Repeat for iteration N

REQUIREMENTS MANAGEMENT

- Establish a requirements change control process
- Perform impact analysis on requirements changes
- Track the status of each requirement; Track requirements issues
- Maintain a history of requirements changes
- Maintain a requirements traceability matrix
- Defining the relationships and dependencies that exist between requirements
- Tracing individual requirements to their corresponding designs, source code, and tests

REQUIREMENTS MANAGEMENT (CNTD.)

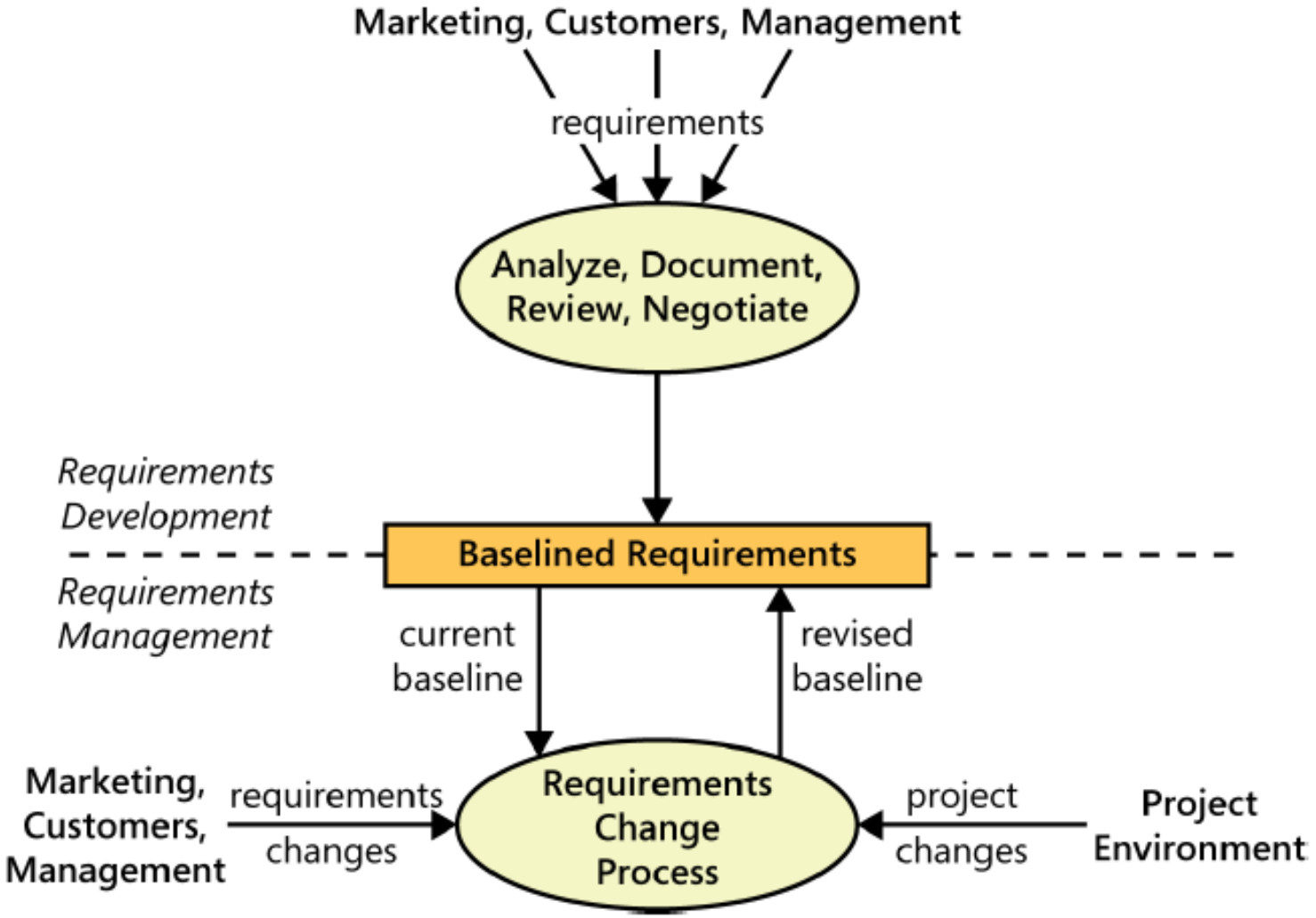


FIGURE 1-5 The boundary between requirements development and requirements management.

A REQUIREMENTS DEVELOPMENT PROCESS FRAMEWORK

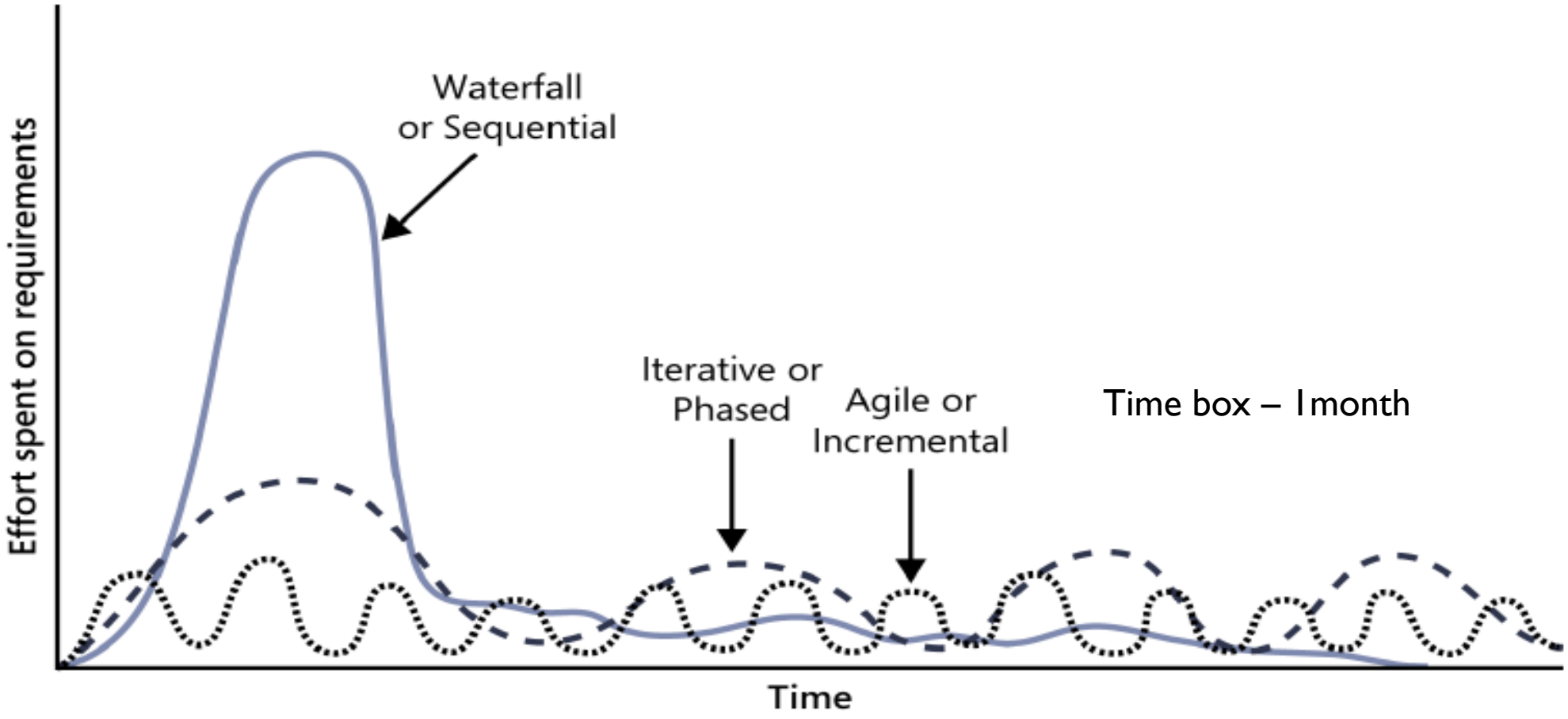


FIGURE 3-3 The distribution of requirements development effort over time varies for projects that follow different development life cycles.

ROLE OF REQUIREMENTS

"The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all the interfaces to people, to machines, and to other software systems. No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later."

REASONS BEHIND BAD REQUIREMENTS

- Insufficient user involvement
- Inaccurate planning
- Creeping (gradually increase, changing) user requirements
- Ambiguous requirements (e.g., response to a request as soon as possible)
- Gold plating (extra functionality beyond the specification)
- Overlooked stakeholders

BENEFITS OF A HIGH-QUALITY REQUIREMENTS 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

REFERENCES

- Wiegers, K., & Beatty, J. (2013). *Software requirements*. Pearson Education.
- http://www.cs.ccsu.edu/~stan/classes/CS530/notes14/04-Requirements.html