ECE 321 Software Requirements Engineering

Lecture 3: Introduction to software requirements

Why do we need requirements?

- **Users want a functional product**
  - They do not want to go back to the developer to ask to add something
- **Developers do not want to learn about new functionality once they deployed the system**
- Solution: a **precise** software requirements document

Software requirements according to the IEEE Standard Glossary of SE

1) A condition or capability needed by a user to solve a problem or achieve an objective

2) A condition or capability that the system must possess to satisfy a contract, standard, or other formal document

3) A documented representation of (1) or (2)

Software requirements according to Sommerville and Sawyer 1997

- Requirements are
  - A specification of what should be implemented
  - Descriptions of how the system should behave
  - A constraint on the development process of the system

More definitions of software requirements

- Anything that drives design choices
  - Lawrence, 1997
- Requirement should be seen as a property that a product must have to provide value to a stakeholder
  - Wiegers, 2003

Are requirements easy to develop?

- No, not at all :(
- There are many people (stakeholders) involved
- Requirements have an enormous impact on the rest of the project
- Wrong requirements are hard to rectify later

Requirements engineering

- An iterative process of
  - Analyzing the problem
  - Documenting the observations
  - Checking the accuracy of the observations
- Concerns functional AND non-functional requirements (performance, reliability, etc.)
- Define **what** will be developed (not **how**)

The involved parties are stakeholders

- Customers
- End-users
- Requirement analysts
- Developers
- Testers
- Documentation writers
- Project managers
- Legal staff
- Manufacturing people

Things to keep in mind about stakeholders

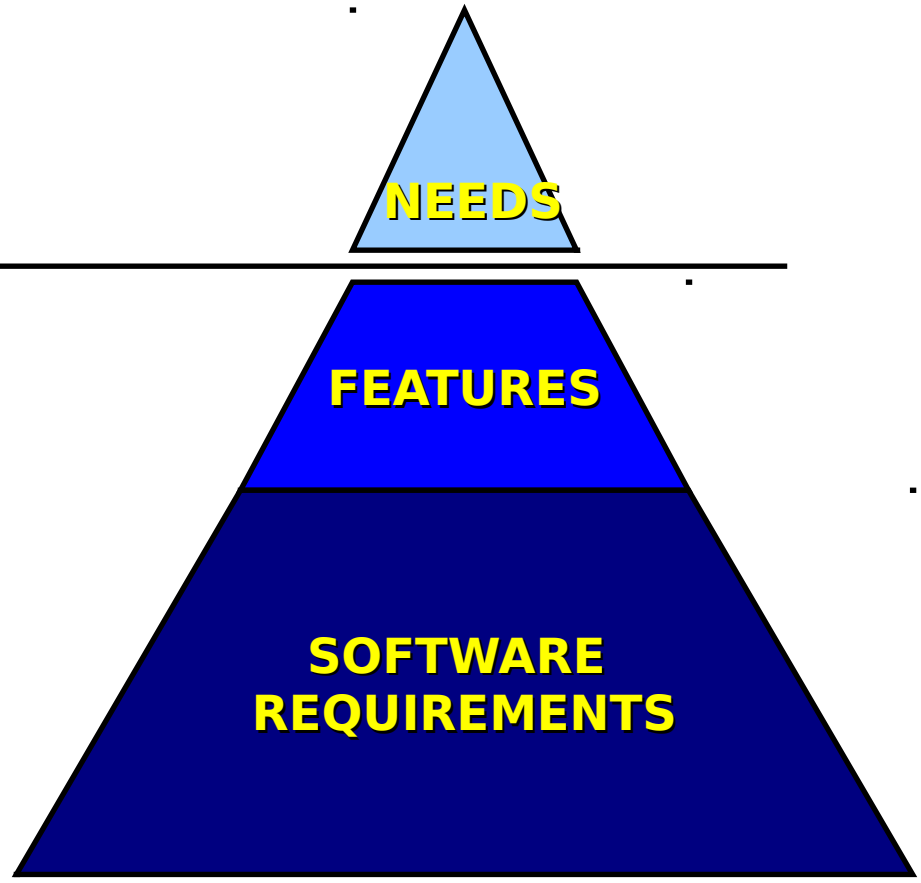
- There can be many!
- They often want conflicting things
  - e.g., red vs. blue button

The goal of software requirements

- To understand stakeholders’ problems in **their** culture and **their** language
- To build systems that meet **their** needs

The software requirements pyramid

- problem domain
- solution domain

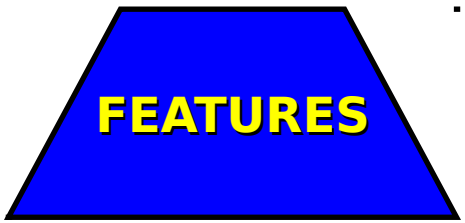


Problem domain

- Understanding the needs of all stakeholders

Solution domain: features

- solution domain
  - What was learned in the problem domain
  - What will be delivered in the solution (definition of services that the system shall provide)



Solution domain: software requirements

- solution domain
  Specific requirements
  that need to be imposed
  on the solution

Proper software requirements

- **Complete** description of functionality
- **Correct** description of functionality
- **Feasible** to implement
- Only define **necessary** requirements

Proper software requirements

- **Prioritize** functionality
- Are **unambiguous**
- Are **verifiable**

Proper requirement specification documents

- Include **all** requirements
- Contain **no conflicts** between requirements
- Are **modifiable** and track changes
- Contain requirements that are **traceable** to their origin, design, source code and test cases

Benefits of good software requirements

- Fewer requirement defects
- Reduced development rework
- Fewer unnecessary features
- Lower enhancement costs
- Faster development
- Fewer miscommunications

Errors in software requirements

- The most common category of system development errors
- Contribute approximately 1/3 of the total delivered defects!
  - Defects that the user sees

How do errors in software requirements happen?

- Insufficient user involvement
- Ambiguous requirements
  - Leave too much room for interpretation by the developer
- Minimal specification
  - Provide concept sketch rather than a complete description

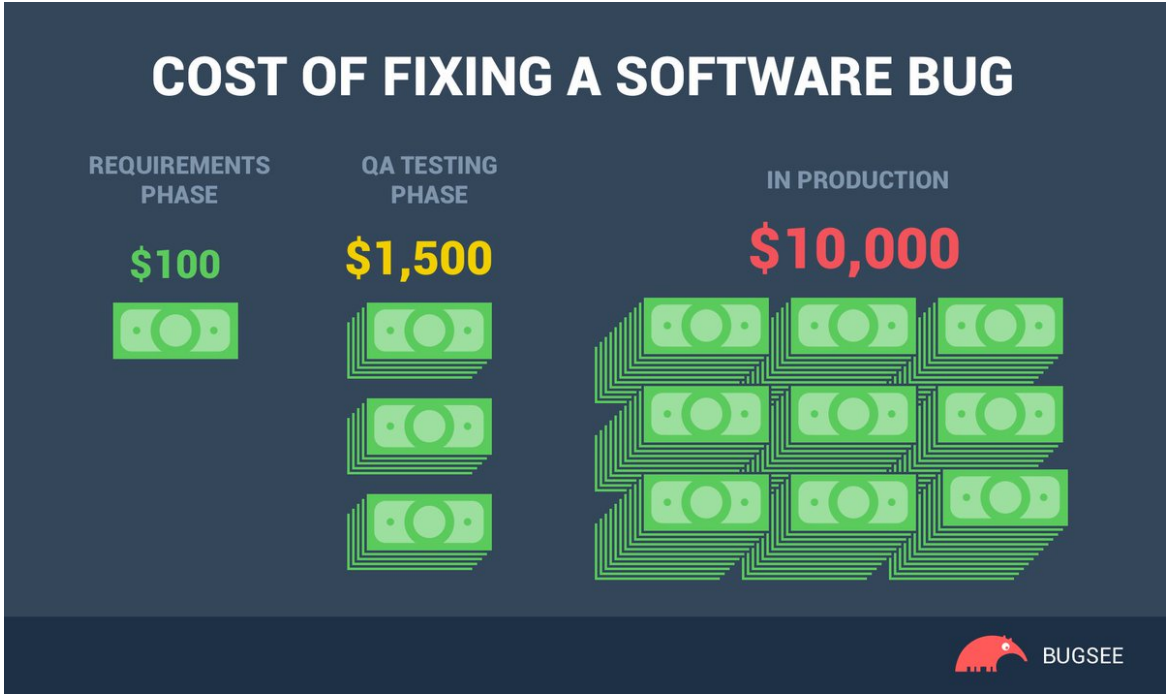
More reasons for errors in software requirements

- Gold plating
  - Functionality that the developer thinks the user will need and like
- Creeping user requirements
  - Many late changes usually result in products that are hard to understand and maintain

Even more reasons for errors in software requirements

- Overlooked user groups
  - Often need to accommodate for different target groups
- Inaccurate planning
  - Vague requirements lead to optimistic cost and effort estimates

Why are errors in requirements so problematic?



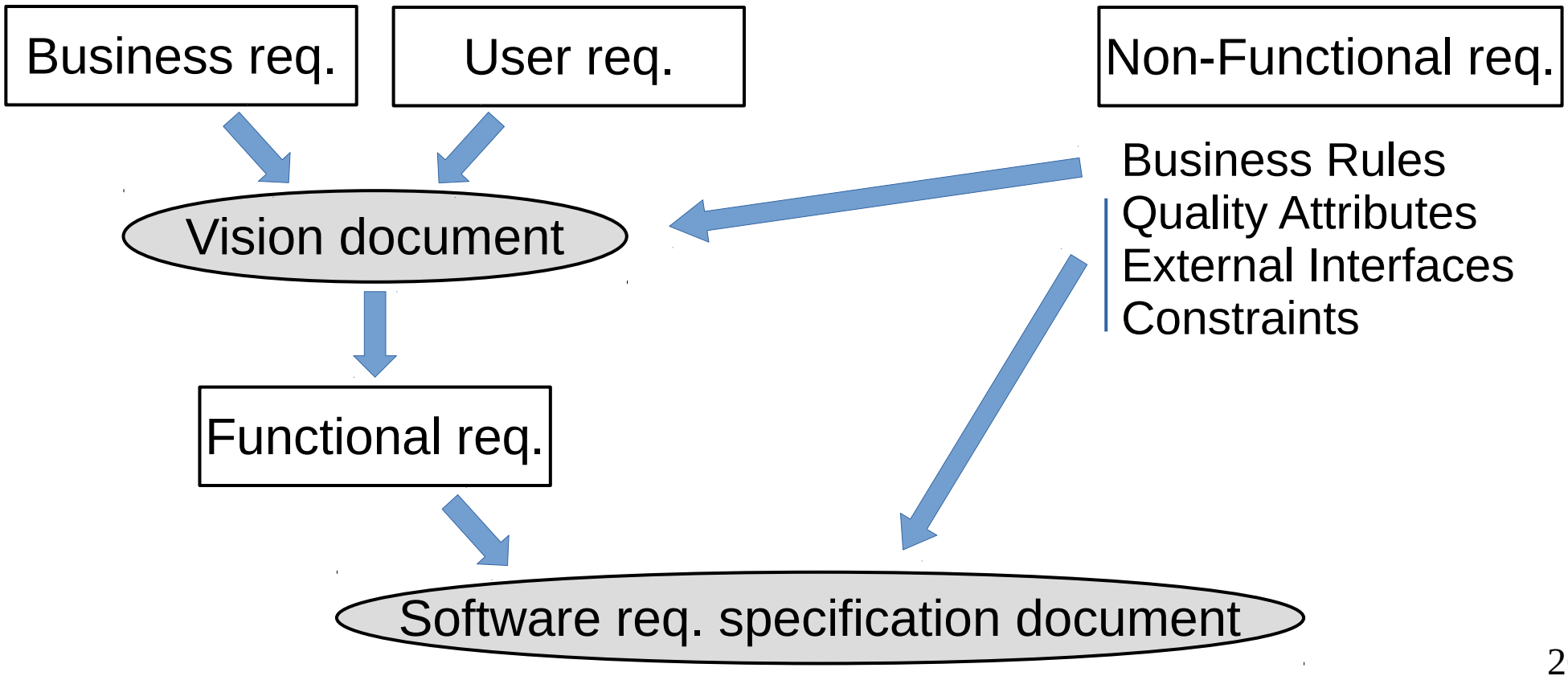
Are problems with requirements common?

- According to a survey with 3,800 responses, the two most common software problems in industry are
  - Requirements specifications
  - Managing customer requirements

Software requirements are captured in two main documents

- **Vision document**
- **Software requirements specification document**
- Four types of requirements
  - Business requirements
  - User requirements
  - Functional requirements
  - Non-functional requirements

The main requirement documents



Business requirements

- High-level objectives of the customer who requests the system
- Define why the system is needed
- What needs to be achieved
- Business rules
  - Policies, regulations, standards etc.

User requirements

- User goals or tasks that users must be able to perform with the product
- Usually defined as use cases and scenario descriptions

Functional requirements

- Functionality that must be built into the product
- Must satisfy the requirements in the vision document
- Provide low-level detailed software functionality
- The traditional ‘shall’ statements
- May contain system requirements
  - Necessary for integrating the developed system in other environments

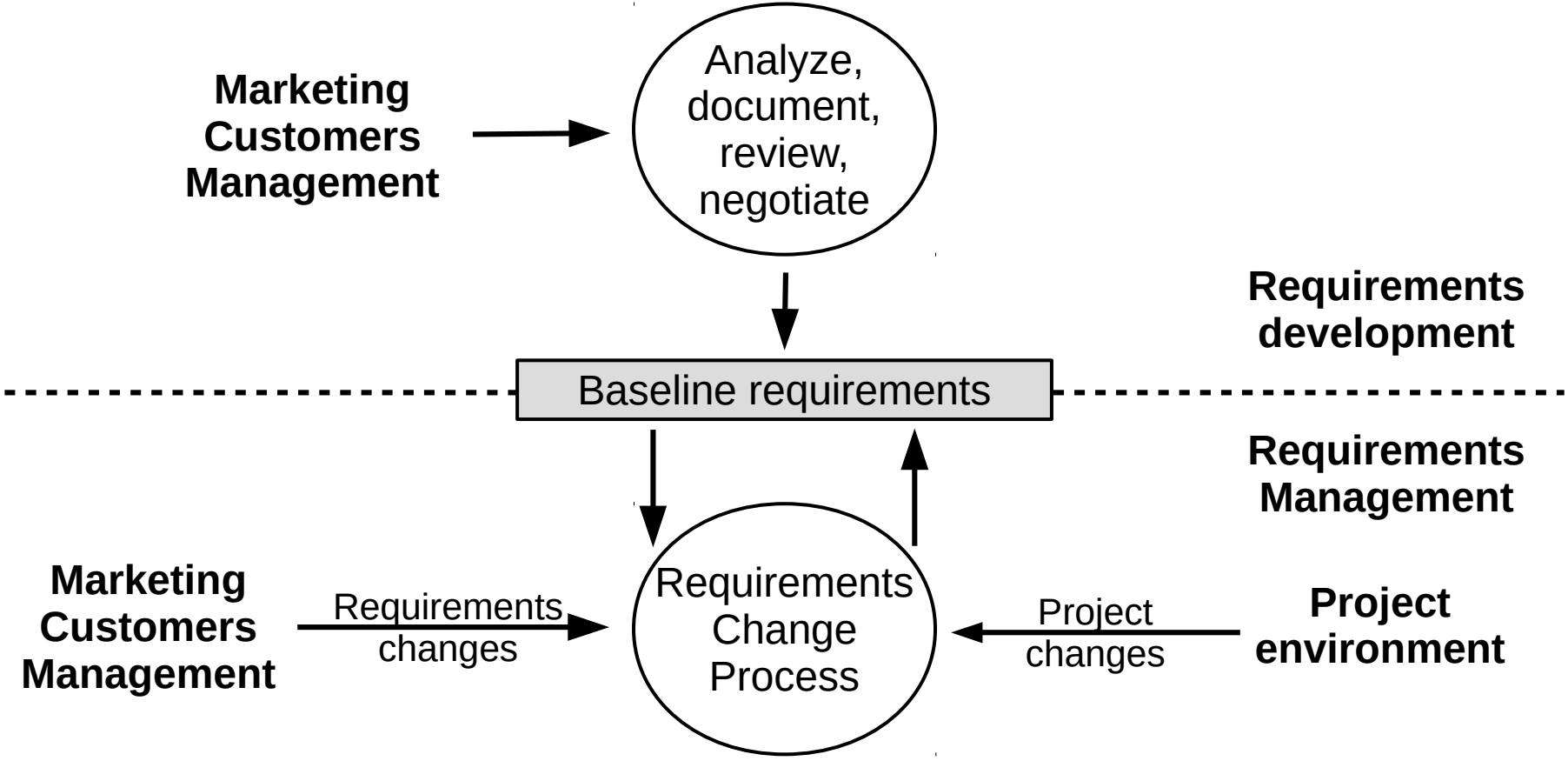
Non-functional requirements

- Quality attributes
  - Product characteristics in various dimensions
    - Portability, reliability, availability, etc.
- External interfaces
  - Connections with rest of the universe (OS, hardware, etc.)
- Constraints
  - Restrictions, e.g., in terms of programming language, tools

Requirements engineering consists of two main phases

- Phase 1: Requirements development
  - Elicitation
  - Analysis
  - Specification
  - Validation
- Phase 2: Requirements management

The phases of requirements engineering



Examples of requirement development tasks

- Identifying user classes for the planned product
- Eliciting needs from individuals
- Understanding users’ tasks, goals and business objectives
- Analyzing the information received from users
- Understanding importance of quality attributes
- Negotiating implementation priorities
- Translating user needs into written documents

Examples of requirement management tasks

- Redefining the requirements baseline
- Reviewing proposed changes and evaluating their impact
- Incorporating approved requirements changes into the project
- Keeping project plans current with the requirements
- Negotiating new commitments
- Tracking individual requirements to their designs, code, tests

The 3 steps of the requirements development process

- **Requirement elicitation**
  - Understanding and analyzing the problem
  - Learning and understanding user needs
- **Requirement specification**
  - Developing a vision document
  - Developing requirement specification document
- **Requirement validation and verification**

Step 1: Requirement elicitation

- Understanding the domain being modelled
  - Interact with stakeholders
  - The analyst is usually not an expert in the domain
  - Usually not only observe, but also model the problem
    - Learn domain based on interviews
    - Analyze the understood knowledge
    - Communicate the knowledge back to the clients to confirm

Step 2: Requirements specification

- Create the formal requirement documents
- Many techniques exist
  - Natural language
    - Informal
    - Formal
  - Language of mathematics
    - Usually mathematical formulation with supporting description in a natural language

Step 3: Requirements validation and verification

- Verify that the correct requirements are stated (validation)
- Verify that the requirements are stated correctly (verification)
- Iterative development of the final requirement specification document

What did we discuss so far?

- What is software requirements engineering
  - Phase 1: requirements development
  - Phase 2: requirements management
- Requirement errors
- Types of requirements