CSSE3012 The Software Process

Week 2: Requirements Engineering

THE UNIVERSITY OF QUEENSLAND
School of Information Technology and Electrical Engineering

Overview

- What is Requirements Engineering (RE)?
- Why is RE important?
- What is a requirement?
- Types of requirements
- What is the requirements process?
- Who does it?

What is Requirements Engineering?

- Systematic approach to acquire, analyse, validate, document and manage requirements
- Typically a cyclic or iterative process
- Requirements validation may include prototype construction and evaluation
- Applied at both system and software levels, often with interleaved system architecture design

What is a Requirement?

1. A condition or capability needed by a 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.

- From IEEE Standard Glossary of Software Engineering Terminology

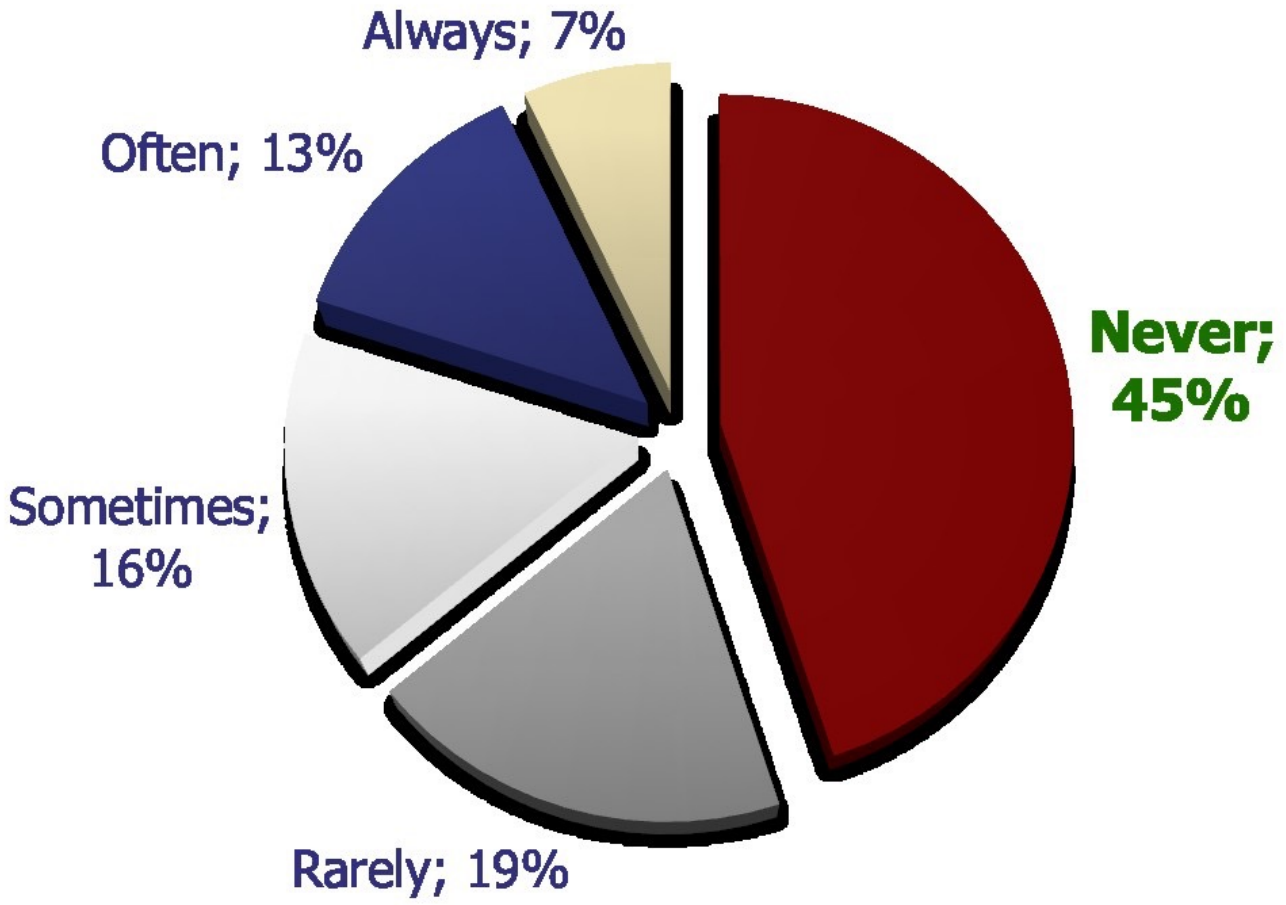
Requirements Engineering Products

- Primary – requirements specification
  - agreement between client and developer
  - basis for all subsequent development and verification processes
- Secondary – usually system and software acceptance test criteria

Why is RE important?

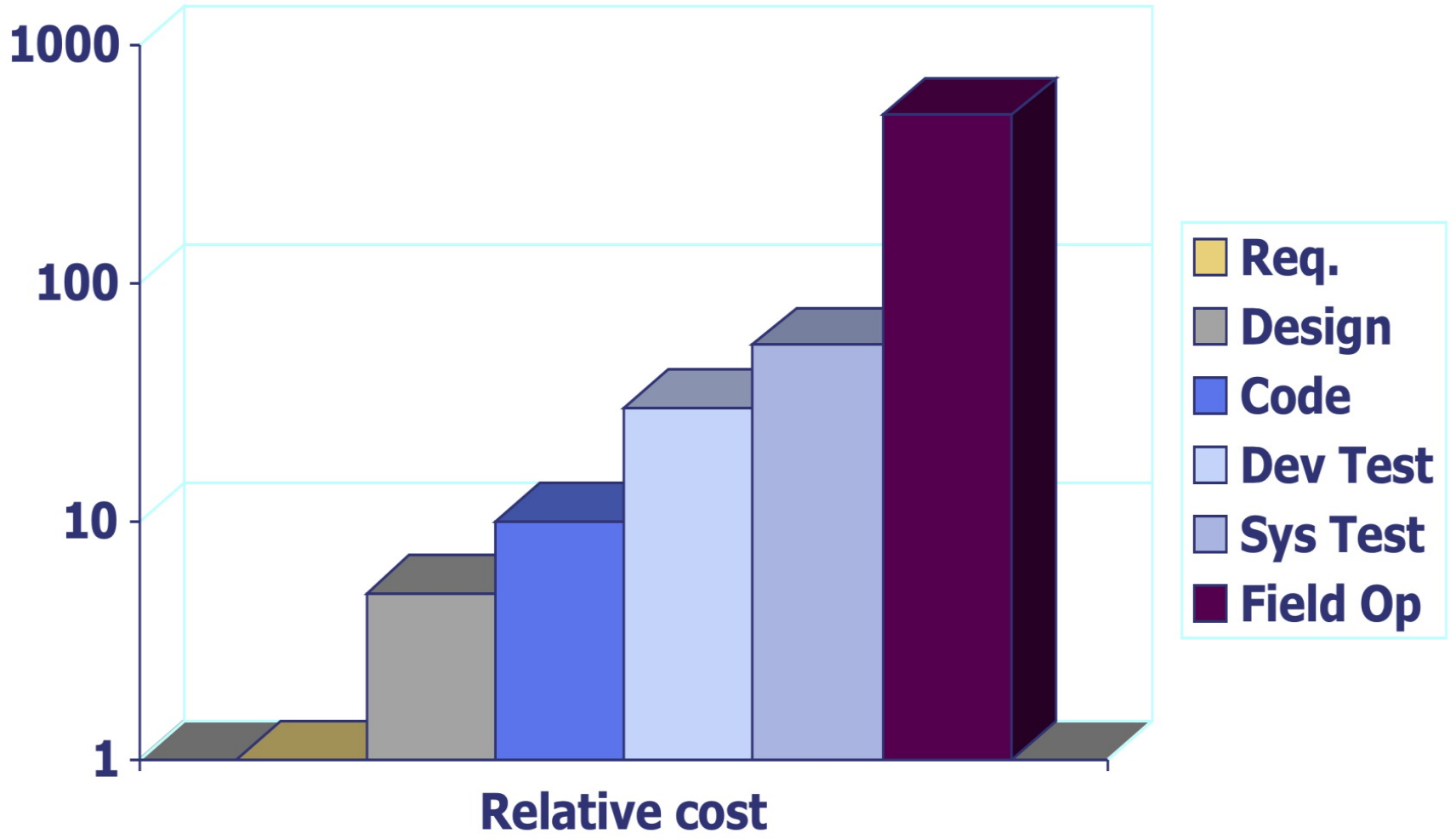
- Most faults observed in a software project are from incorrect, incomplete, or misinterpreted functional specifications or requirements.
  - Glass' Law
- "On average, **40%** of the requirements specified in the Feasibility and Requirements phase of the lifecycle were redefined in the subsequent four *lifecycle phases… on average Digital spent 50% more* than budgeted."
  - Hutchings and S. Knox: "Creating products customers demand", *Communications of the ACM*, May 1995

Use of Waterfall-Specified Features



Johnson, J. ROI – It's Your Job. XP 2002.

Relative Cost to Correct Requirement Faults



Average Software Quality Costs

Defect Origins	Defect Potential	Removal Efficiency	Delivered Defects
Requirements	1.00	77%	0.23
Design	1.25	85%	0.19
Coding	1.75	95%	0.09
Documents	0.60	80%	0.12
Bad Fixes	0.40	70%	0.12
Total	5.00	85%	0.75

- Defects per function point
- Capers Jones, 2012

Best in Class Software Quality Costs

Defect Origins	Defect Potential	Removal Efficiency	Delivered Defects
Requirements	0.40	85%	0.08
Design	0.60	97%	0.02
Coding	1.00	99%	0.01
Documents	0.40	98%	0.01
Bad Fixes	0.10	95%	0.01
Total	2.50	96%	0.13

- Defects per function point
- Most often observed in CMM level 3 or TSP projects
- Capers Jones, 2012

Poor Software Quality Costs

Defect Origins	Defect Potential	Removal Efficiency	Delivered Defects
Requirements	1.50	50%	0.75
Design	2.20	50%	1.10
Coding	2.50	80%	0.50
Documents	1.00	70%	0.30
Bad Fixes	0.80	50%	0.40
Total	8.00	62%	3.05

- Defects per function point
- Most often observed in large plan-driven projects
  - > 10K function points
- Capers Jones, 2012

Why is RE important? (Cont.)

- Helps earlier detection of mistakes, which are much more costly to correct if discovered later
- Forces clients to articulate and review requirements
- Enhances communications between participants
- Helps record and refine requirements
- All this is about producing good requirements!

What Happens if the Requirements are Wrong?

- Delivered late
- Over budget
- Customers are not satisfied
  - may not use its facilities
  - may decide to scrap it
- System may be unreliable
- Maintenance and extension may be very costly

Benefits of Good Requirements

- Agreement among developers, customers and users
  - on what is to be done
  - acceptance criteria for the delivered system
- Sound basis for resource estimation
  - cost, personnel quantity and skills, equipment and time
- Improved system usability, maintainability and other quality attributes
- Achievement of goals with minimum resources
  - less rework, fewer omissions and misunderstandings

Advice / Perspective

- … a systematic approach to finding, documenting, organizing, and tracking the **changing** requirements of a system [RUP]
- In practice it is impossible to produce a complete and consistent requirements document [Somerville]
- Getting the requirements right is critical for success
- Requirements are rarely right at the start of a large project
  - expect change
  - manage it
  - agile mantra “Embrace Change”

Functional Requirements

- Requirements (or capabilities) for functions (specific behaviour) that must be performed by the system
  - e.g. read a bar code, change a user name, maintain a temperature
- Primary focus of most requirements activities

Non-Functional Requirements

- Constraints on performance or quality
- **Product Properties**
  - Requirements on the behaviour of the product
    - System shall process a minimum of 8 transactions per second
    - User credit card details shall be secured
- **Process Properties**
  - Requirements on the practices used to develop / produce the system
    - Control software shall be verified in accordance with IEEE STD 1012-2016

Requirements (Cont.)

- Relationship between quality / cost / timeliness / etc. of the product and the quality of the process
- Both functional and non-functional requirements are essential for successful software
  - both must be verified
    - consequently they must be testable
      - Non-Functional Requirements (NFR) should be measurable

Classification of Non-Functional Req.

According to the ISO standard

- Safety requirements
- Security requirements
- Interface requirements
- Human engineering requirements
- Qualification requirements
- Operational requirements
- Maintenance requirements
- Design constraints

Non-Functional Requirements Examples

- Safety Requirements
  - The system shall not permit operation unless the operator guard is in place.
- Security Requirements
  - Only the system administrator can change system data.
  - All system data must be backed up every 24 hours.
- Interface Requirements
  - Interaction with other existing or proposed systems
    - e.g. specific databases, API's for other systems

Non-Functional Req. Examples (cont.)

- Human Engineering Requirements (usability)
  - Adequacy requirements – system does what is required
  - Learning requirements – time needed to learn the facilities of the system
  - (Error) handling requirements – error rate of the end-users
  - Recovering requirements – time to restart after system failure

Non-Functional Req. Examples (cont.)

- Qualification Requirements (set V&V targets)
  - Queensland government requires certification to the quality standard ISO 9001 (quality management systems) for its major software suppliers
- Operational Requirements
  - (Components of) an industrial or military control system may have to withstand extreme heat or cold, to survive vibration, movement, sudden impact, etc.
  - System efficiency or performance
    - limits on memory and processor speed are often consequences of the operational environment

Non-Functional Req. Examples (cont.)

- Maintenance Requirements
  - Software readability, flexibility/portability and testability
- Design Constraints
  - Use database X because of user familiarity
  - Use algorithm Y for function Z because of stakeholder preference

Sources of Requirements

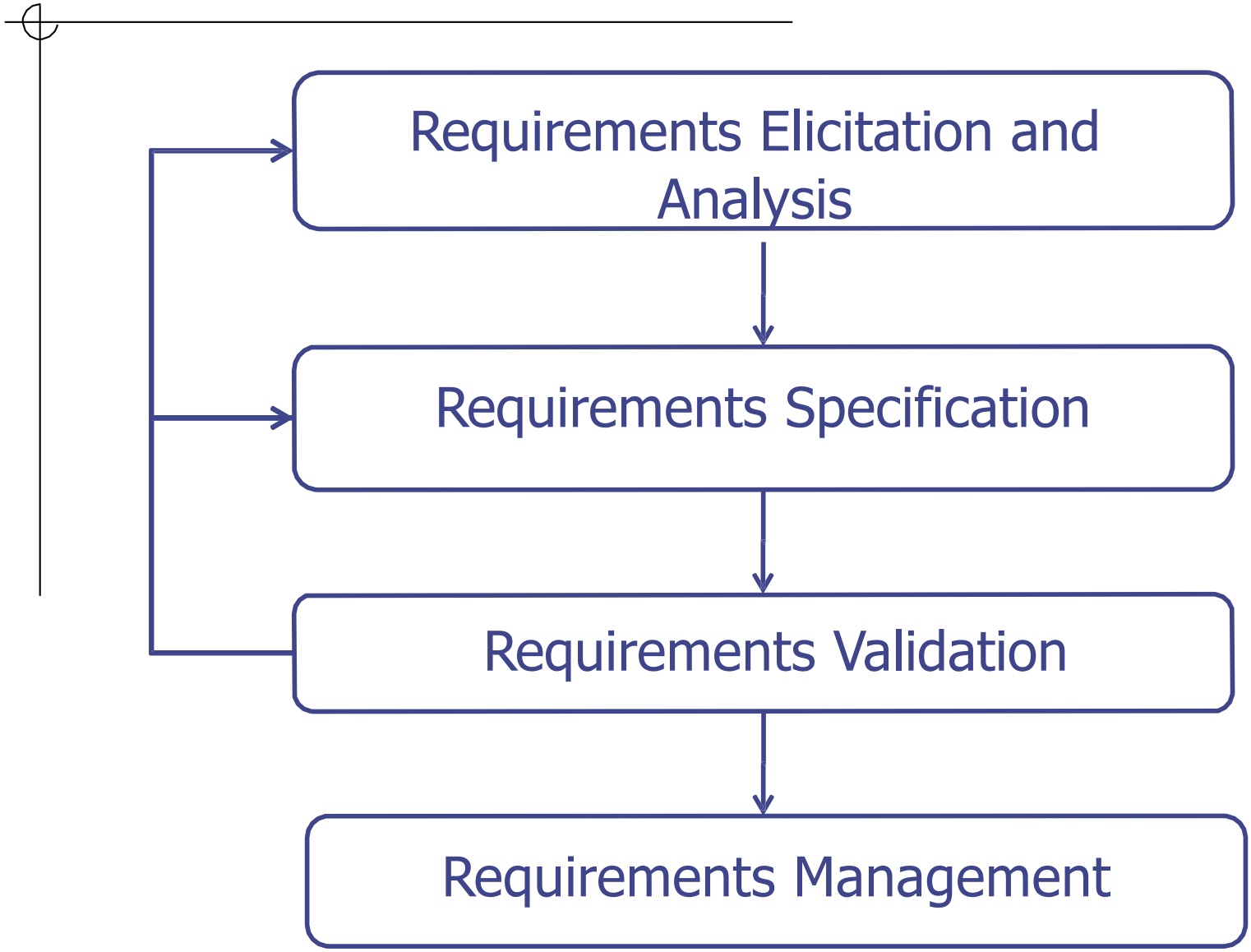
- Various Stakeholders

Users
e.g. customers or end-users – user requirements

Other Stakeholders
e.g. marketing experts, regulators, managers, business owners, developers

Non-Human Sources
e.g. other devices or systems in the environment

Requirements Engineering Process



Who Does It?

User organisation (Domain competent)

Universities prepare requirements specifications for student enrolment systems (SI-net)

Developer organisation (IT competent)

IT companies (e.g. Technology One) prepare requirements specifications for their clients

Third party organisation (Domain and IT competent)

Consulting companies (e.g. Accenture) prepare requirements specifications for their clients

Summary

- Requirements engineering is concerned with eliciting, analysing, documenting, validating and managing requirements.
- RE process, and the specification it produces, is the primary link between the user and the system developer.
- Requirements specification is basis for all subsequent development activity.
  - must guide decisions that determine product quality
- Requirements engineering is key to product quality.

Reading

- Wiegers – chapter 1
- Sommerville – chapter 4.1 & 4.2