Requirements Engineering

Department of Software Engineering
College of Information & Communication Technology
Cantho University

Software-Intensive Systems

- **Software (on its own) is useless**
  - Software is a set of computations
  - Software only becomes useful when run on some hardware
  - Software + Hardware = “Computer System”
- **A Computer System (on its own) is useless**
  - Only useful in the context of some human activity that it can support
  - Software + Hardware + Human Activities = “Software-Intensive System”

Quality = Fitness for purpose

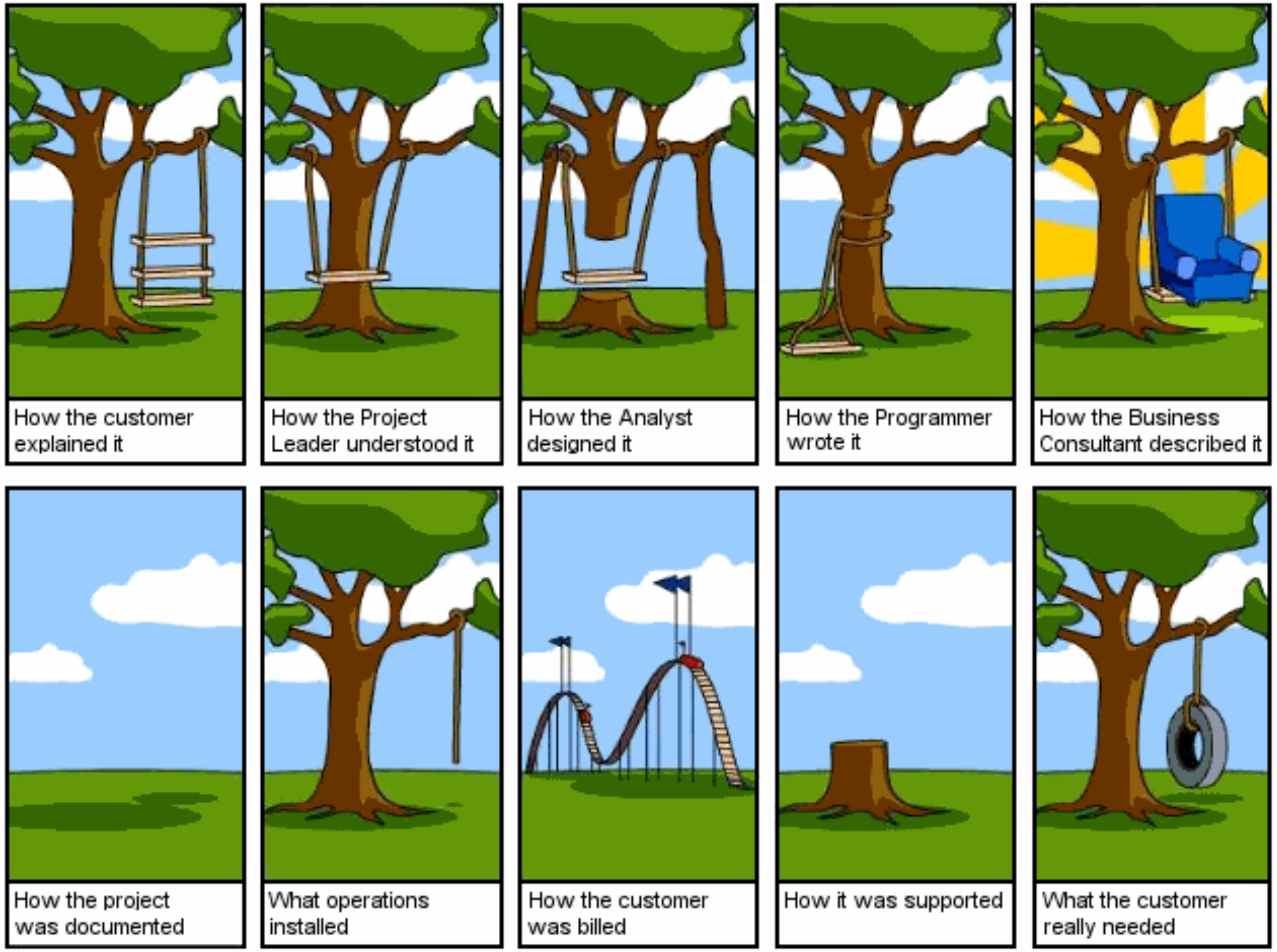
- **Software is designed for a purpose**
  - If it doesn’t work well then either:
    - The designer didn’t have an adequate understanding of the purpose
    - or we are using the software for a purpose different from the intended one
  - Requirements analysis is about identifying this purpose
  - Inadequate or wrong understanding of the purpose leads to poor quality software

Quality = Fitness for purpose (cont.)

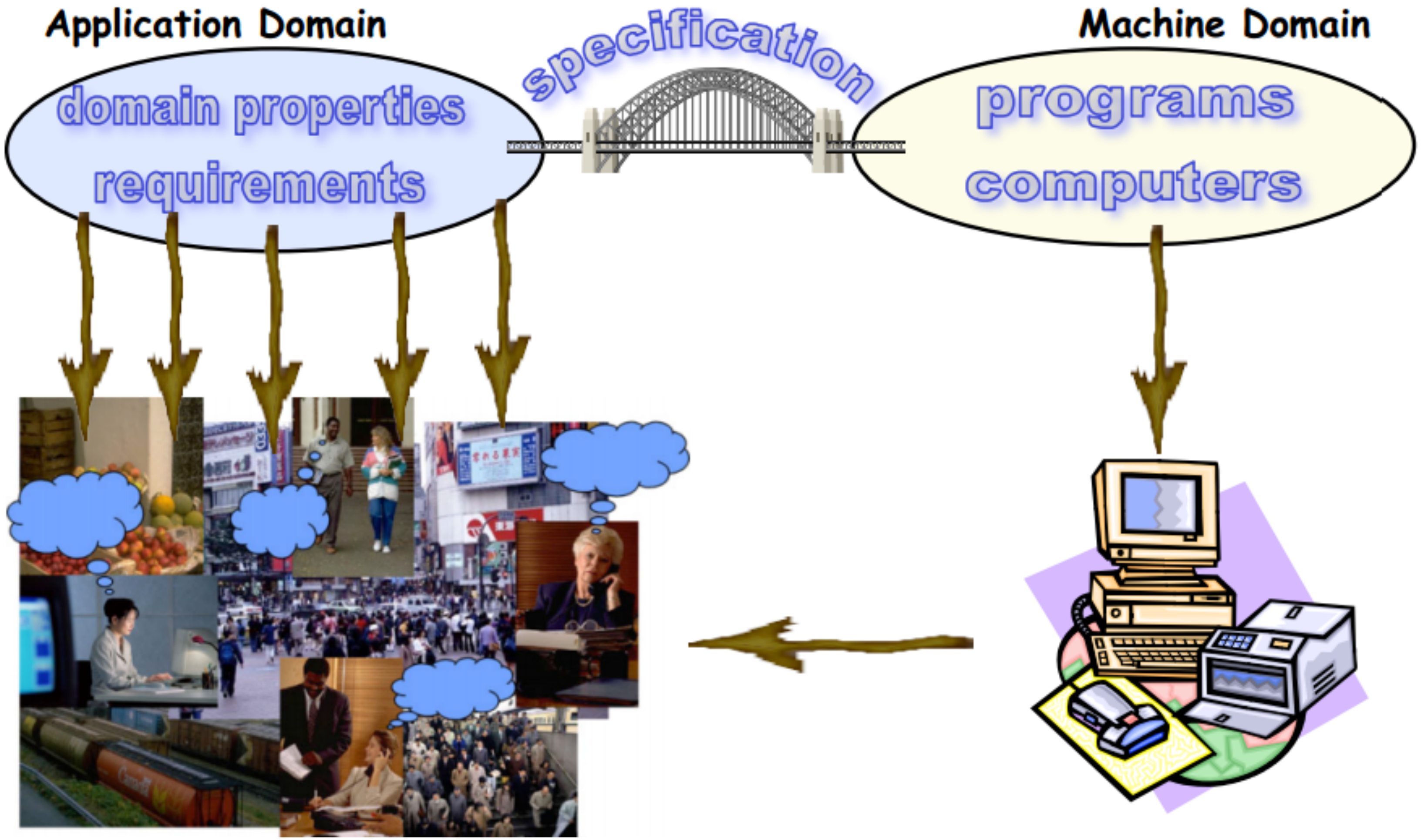
- The purpose is found in human activities
  - E.g. Purpose of a student management system comes from the management activities of university and the needs of managers.

- The purpose is often complex:
  - Many different kinds of people and activities.
  - Conflicting interests among them.

Quality = Fitness for purpose (cont.)


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Where are the challenges?



Which systems are soft?

- **Generic software components**
  - E.g. Core operating system functions, network services, …
  - Functionality relatively stable, determined by technical interfaces.
  - But note that these systems still affect human activity.
    - E.g. concepts of a ‘URL’, etc.

Which systems are soft? (cont.)

- Control Systems
  - E.g. aircraft flight control, industrial process control, …
  - Most requirements determined by the physical processes to be controlled.
  - But note that operator interaction is usually crucial
    - E.g. accidents caused when the system doesn't behave as the operator expected.

(1996, tên lửa Ariane 5)

Which systems are soft? (cont.)

- **Information Systems**
  - E.g. office automation, web services, business support,…
  - These systems cannot be decoupled from the activities they support.
  - Design of the software entails design of the human activity.
    - The software and the human activities co-evolve

Definition of Requirement

- 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.

Sommerville and Pete Sawyer (1997)

Levels and types of requirements

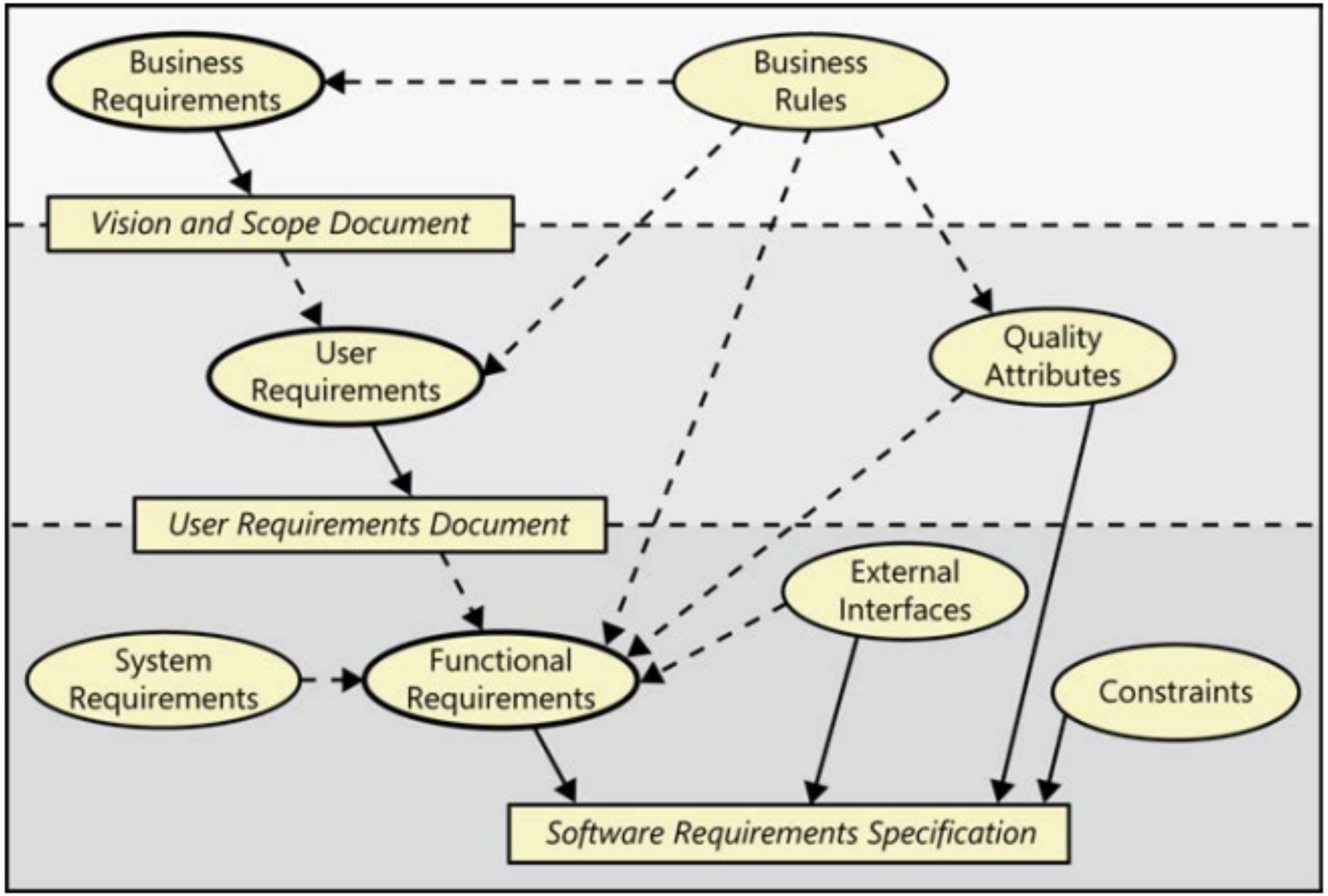
Term	Definition
Business requirement	A high-level business objective of the organization that builds a product or of a customer who procures it.
Business rule	A policy, guideline, standard, or regulation that defines or constrains some aspect of the business. Not a software requirement in itself, but 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.

Levels and types of requirements (cont.)

Term	Definition
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.

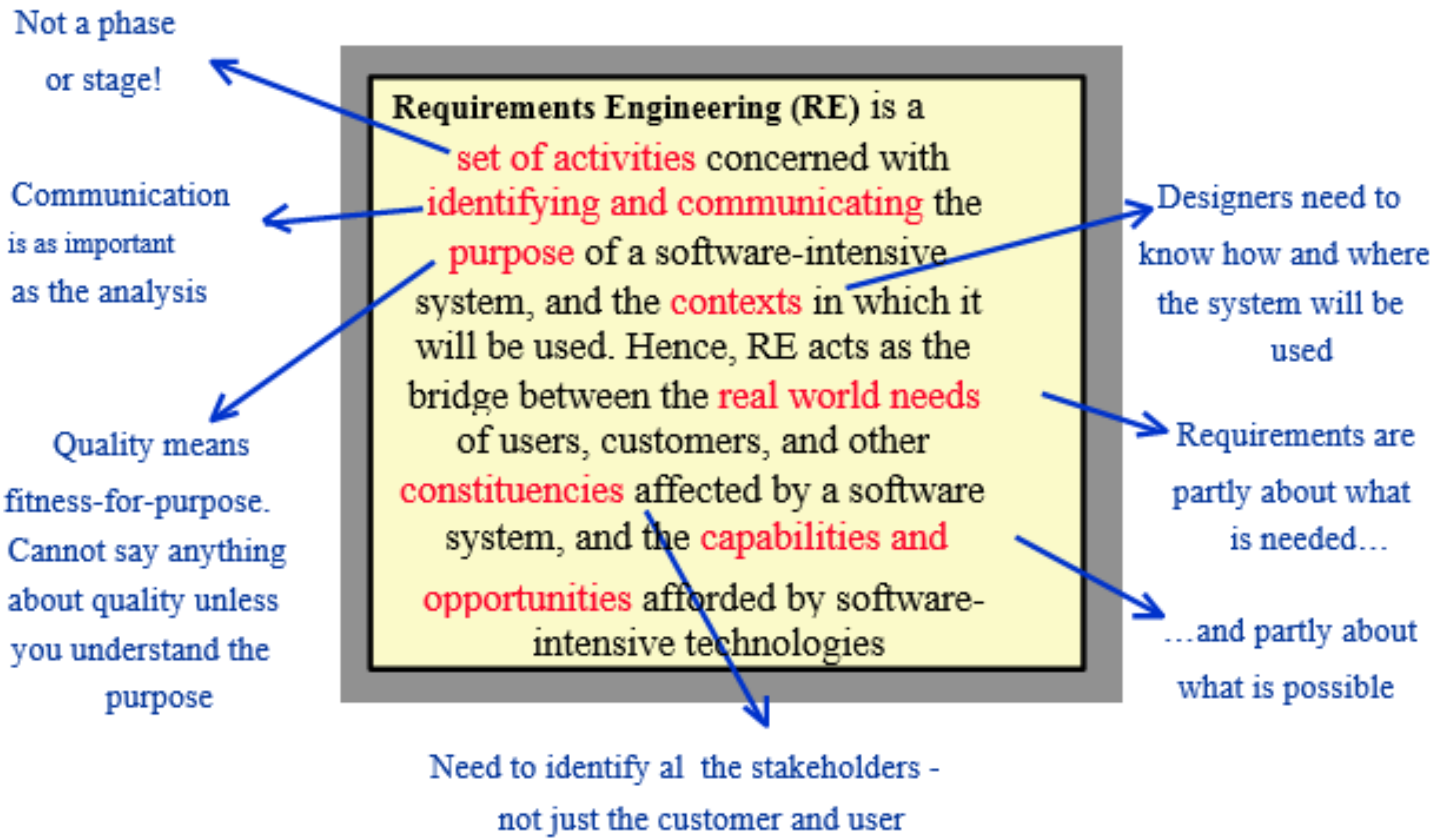
- Software requirements include three distinct levels: business requirements, user requirements, and functional requirements.

Levels and types of requirements (cont.)



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

Definition of RE



Requirements Engineering (RE) is a set of activities concerned with identifying and communicating the purpose of a software-intensive system, and the contexts in which it will be used. Hence, RE acts as the bridge between the real world needs of users, customers, and other constituencies affected by a software system, and the capabilities and opportunities afforded by software-intensive technologies

Not a phase or stage!

Communication is as important as the analysis

Quality means fitness-for-purpose. Cannot say anything about quality unless you understand the purpose

Designers need to know how and where the system will be used

Requirements are partly about what is needed…

…and partly about what is possible

Need to identify al the stakeholders - not just the customer and user

Cost of getting it wrong

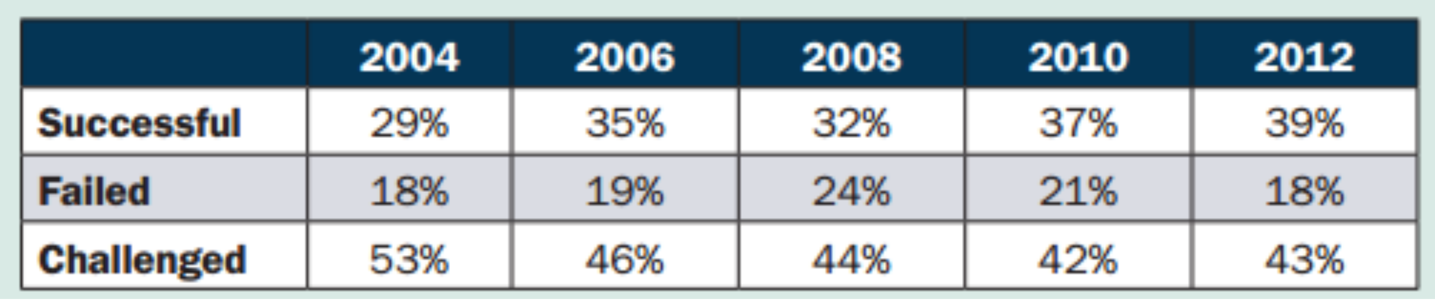
- Cost of fixing errors
  - Typical development process

requirements analysis ⇒ software design ⇒ programming ⇒ development testing ⇒acceptance testing ⇒ operation

  - Errors cost more to fix the longer they are undetected
    - E.g. A requirements error found in testing costs 100 times more than a programming error found in testing

Cost of getting it wrong (cont.)

- Causes of project failure
  - Survey of US software projects by the Standish group:



	2004	2006	2008	2010	2012
Successful	29%	35%	32%	37%	39%
Failed	18%	19%	24%	21%	18%
Challenged	53%	46%	44%	42%	43%

Errors introduced during requirements activities account for 40 to 50 percent of all defects found in a software product (Davis 2005).

Causes of fails:

1. Large problems
2. Lack of training in software engineering
3. Objectives are unclear
4. Requirement specifications are uncompleted, wrong
5. Changing requirements
6. Errors in designing and implementing phases
7. Lack of plans

Cost of getting it wrong (cont.)

- It is very cost if errors can’t early detected in the development process
- Boehm and Papaccio (1988) estimated:

requirements analysis ($1) ⇒ software design ($5) ⇒ programming ($10) ⇒ development testing ($20) ⇒ operation ($100)

- It is necessary to gather and analysis requirements carefully.

What do Requirements Analysts do?

- Starting point
  - Some notion that there is a “problem” that needs solving
    - e.g. dissatisfaction with the current state of affairs
    - e.g. a new business opportunity
    - e.g. a potential saving of cost, time, resource usage, etc.
  - A Requirements Analyst is an agent of change

What do Requirements Analysts do? (cont.)

- The requirements analyst must:
  - Identify the “problem”/”opportunity”
    - Which problem needs to be solved? (identify problem Boundaries)
    - Where is the problem? (understand the Context/Problem Domain)
    - Whose problem is it? (identify Stakeholders)
    - Why does it need solving? (identify the stakeholders’ Goals)
    - How might a software system help? (collect some Scenarios)
    - When does it need solving? (identify Development Constraints)
    - What might prevent us solving it? (identify Feasibility and Risk)

Some observations about RE

- RE is not necessarily a sequential process
  - Don’t have to write the problem statement before the solution statement.
    - (Re-)writing a problem statement can be useful at any stage of development
- The problem statement will be imperfect
  - RE models are approximations of the world
    - will contain inaccuracies and inconsistencies
    - will omit some information
    - analysis should reduce the risk that will cause serious problems…

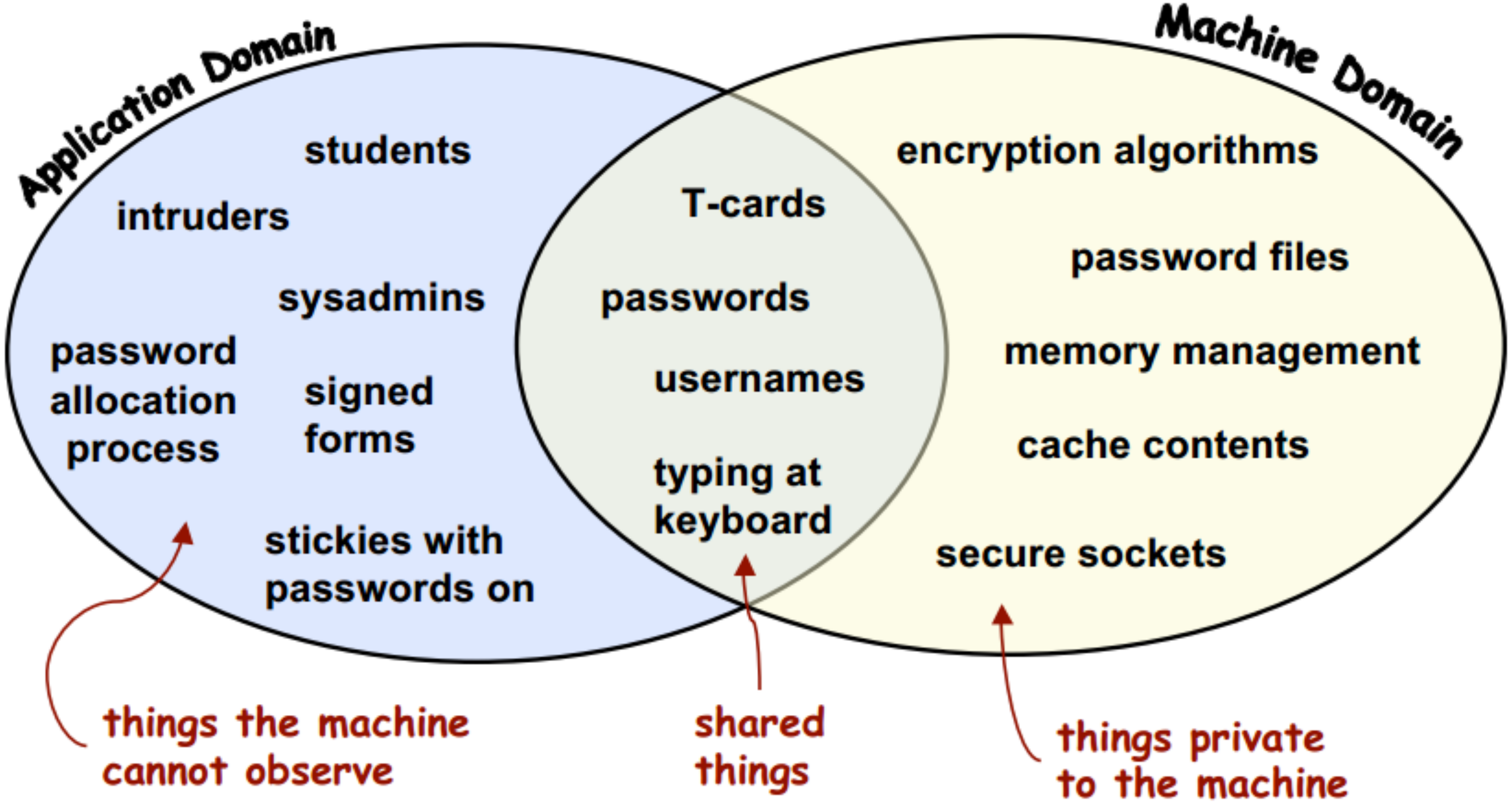
Some observations about RE (cont.)

- Perfecting a specification may not be cost-effective
  - Requirements analysis has a cost
  - For different projects, the cost-benefit balance will be different

- Problem statement should never be treated as fixed
  - Change is inevitable, and therefore must be planned for

Describing a problem

- E.g. “prevent unauthorized access to machines”



What are requirements?

- Domain Properties
  - Things in the application domain that are true whether or not we ever build the proposed system.
- Requirements
  - Things in the application domain that we wish to be made true by delivering the proposed system.
    - Many of which will involve phenomena the machine has no access to
- A Specification
  - is a description of the behaviours that the program must have in order to meet the requirements.

Fitness for purpose?

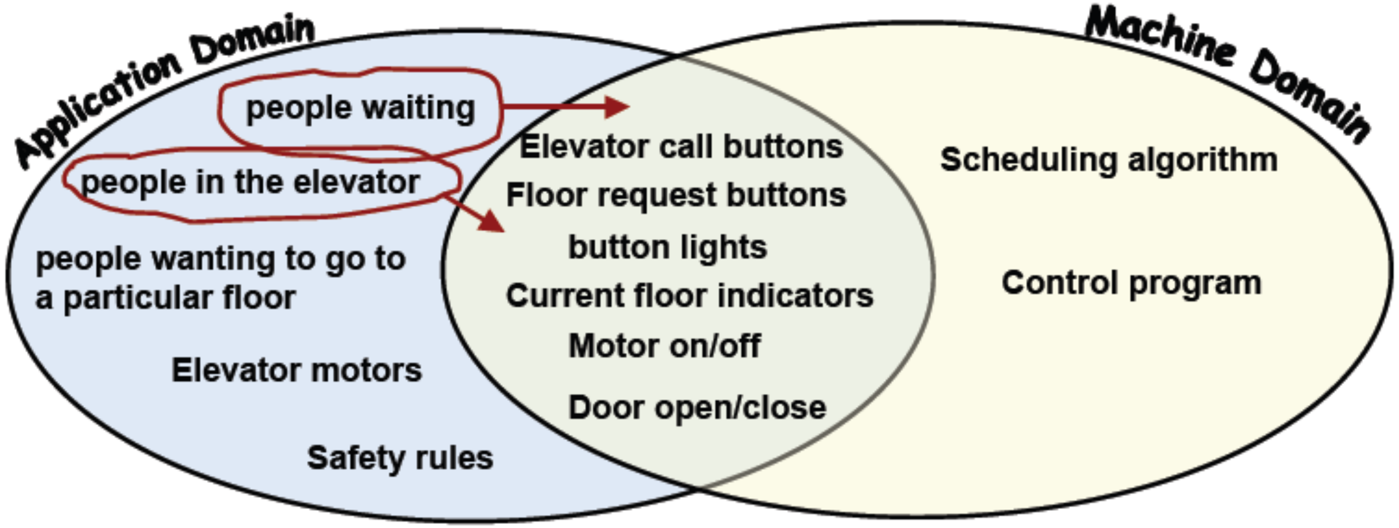
- Two correctness (verification) criteria
  - The Program running on a particular Computer satisfies the Specification.
  - The Specification, in the context of the given domain properties, satisfies the requirements.
- Two completeness (validation) criteria
  - We discovered all the important requirements.
  - We discovered all the relevant domain properties.

Fitness for purpose? (cont.)

- Example
  - Requirement R:
    - “Reverse thrust shall only be enabled when the aircraft is moving on the runway”.
  - Domain Properties D:
    - Wheel pulses on if and only if wheels turning.
    - Wheels turning if and only if moving on runway.
  - Specification S:
    - Reverse thrust enabled if and only if wheel pulses on.
  - Verification: S, D ⇨R
  - Validation:
    - Are our assumptions about the domain correct?
    - Is the requirement really important?

But we can also move the boundaries…

- E.g. Elevator control system:



- We can shift things around:
  - E.g. Add some sensors to detect when people are waiting
  - This changes the nature of the problem to be solved

What is engineering?

"Engineering is the development of **cost-effective** **solutions** to **practical problems**, through the application of scientific knowledge"

- **"…cost-effective…"**
  - Consideration of design trade-offs, resource usage.
  - Minimize negative impacts (e.g. environmental and social cost)
- **"…solutions…"**
  - Emphasis on building devices.
- **"…practical problems…"**
  - solving problems that matter to people.
  - improving human life in general through technological advance.

Project Management

- A manager can control 4 things:
  - Resources (can get more dollars, facilities, personnel)
  - Time (can increase schedule, delay milestones, etc.)
  - Product (can reduce functionality)
  - Risk (can decide which risks are acceptable)
- To do this, a manager needs to keep track of:
  - Effort - How much effort will be needed? How much has been expended?
  - Time - What is the expected schedule? How far are we deviating from it?
  - Size - How big is the planned system? How much have we built?
  - Defects - How many errors are we making? How many are we detecting?
    - And how do these errors impact quality?

Project Management (cont.)

- Requirements are the foundation for both the software development and the project management activities.
- Initially, a manager needs good estimates
  - …and these can only come from a thorough analysis of the problem.

You cannot control that which you cannot measure!

Project Types

- Reasons for initiating a software development project
  - Problem-driven: competition, crisis,…
  - Change-driven: new needs, growth, change in business or environment,…
  - Opportunity-driven: exploit a new technology,…
  - Legacy-driven: part of a previous plan, unfinished work, …

Project Types (cont.)

- Relationship with Customer(s):
  - Customer-specific - one customer with specific problem
    - May be another company, with contractual arrangement
    - May be a division within the same company
  - Market-based - system to be sold to a general market
    - In some cases the product must generate customers
    - Marketing team may act as substitute customer
  - Community-based - intended as a general benefit to some community
    - E.g. open source tools, tools for scientific research
  - Hybrid (a mix of the above)

Lifecycle of an Engineering Project

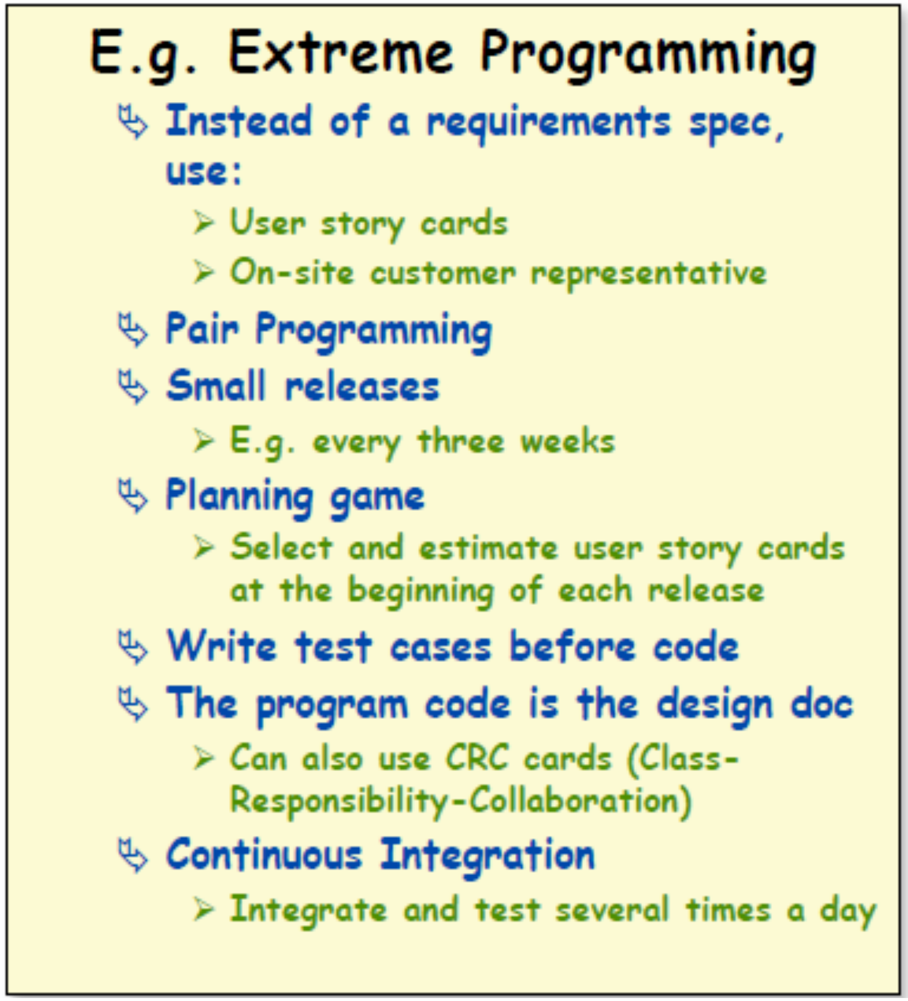
- Examples:
  - Sequential models: Waterfall, V model
  - Rapid Prototyping
  - Phased Models: Incremental, Evolutionary
  - Iterative Models: Spiral
  - Agile Models: eXtreme Programming

Agile Models

- Basic Philosophy
  - Reduce communication barriers
    - Programmer interacts with customer
  - Reduce document-heavy approach
    - Documentation is expensive and of limited use
  - Have faith in the people
    - Don’t need process models to tell them what to do!
  - Respond to the customer
    - Rather than focusing on the contract

Agile Models (cont.)

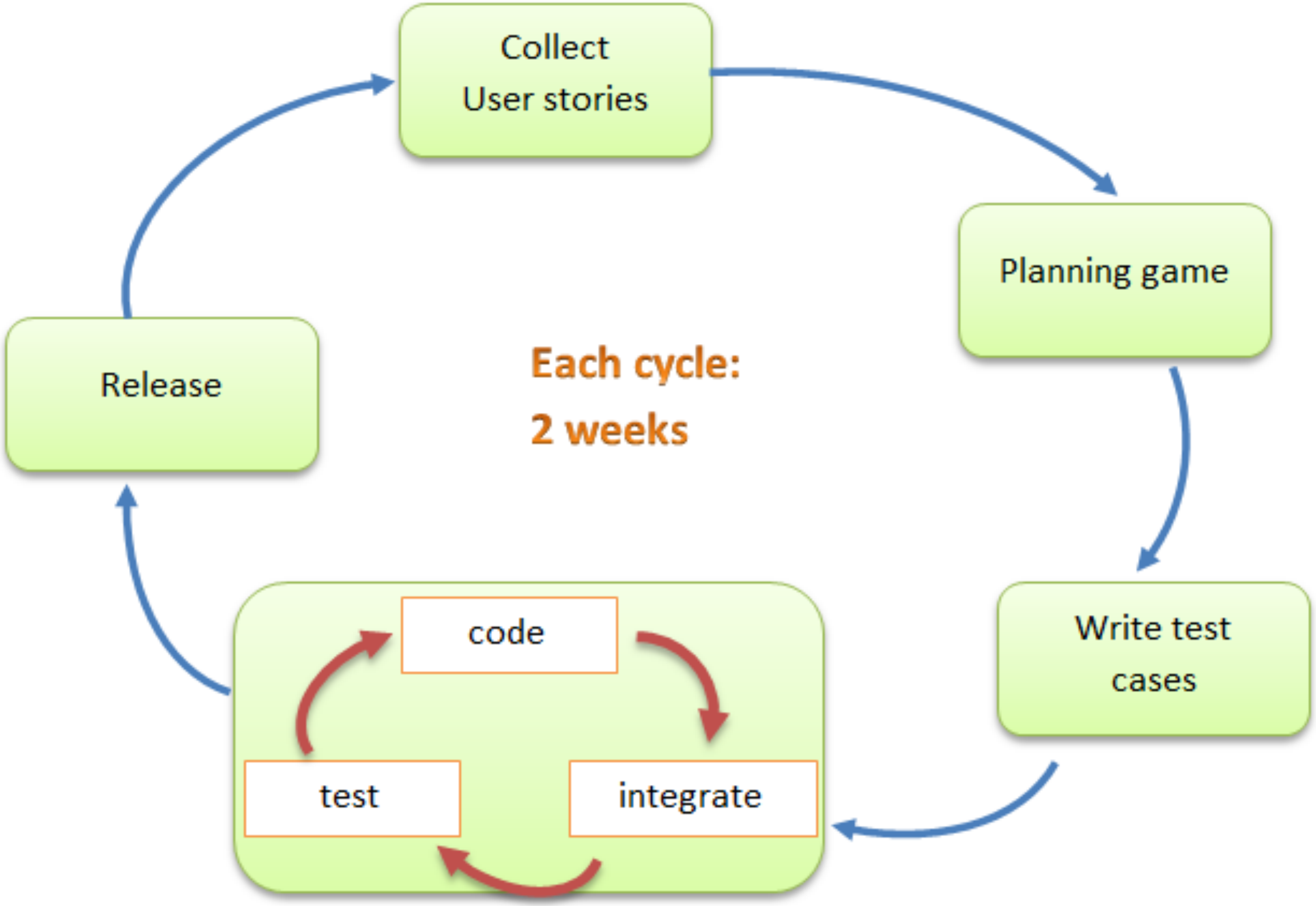
- Weaknesses
  - Relies on programmer’s memory
    - Code can be hard to maintain
  - Relies on oral communication
    - Mis-interpretation possible
  - Assumes single customer representative
    - Multiple viewpoints not possible
  - Only short term planning
    - No longer term vision



E.g. Extreme Programming

- Instead of a requirements spec, use:
  - User story cards
  - On-site customer representative
- Pair Programming
- Small releases
  - E.g. every three weeks
- Planning game
  - Select and estimate user story cards at the beginning of each release
- Write test cases before code
- The program code is the design doc
  - Can also use CRC cards (Class-Responsibility-Collaboration)
- Continuous Integration
  - Integrate and test several times a day

Agile Models (cont.)



Collect User stories

Planning game

Write test cases

code

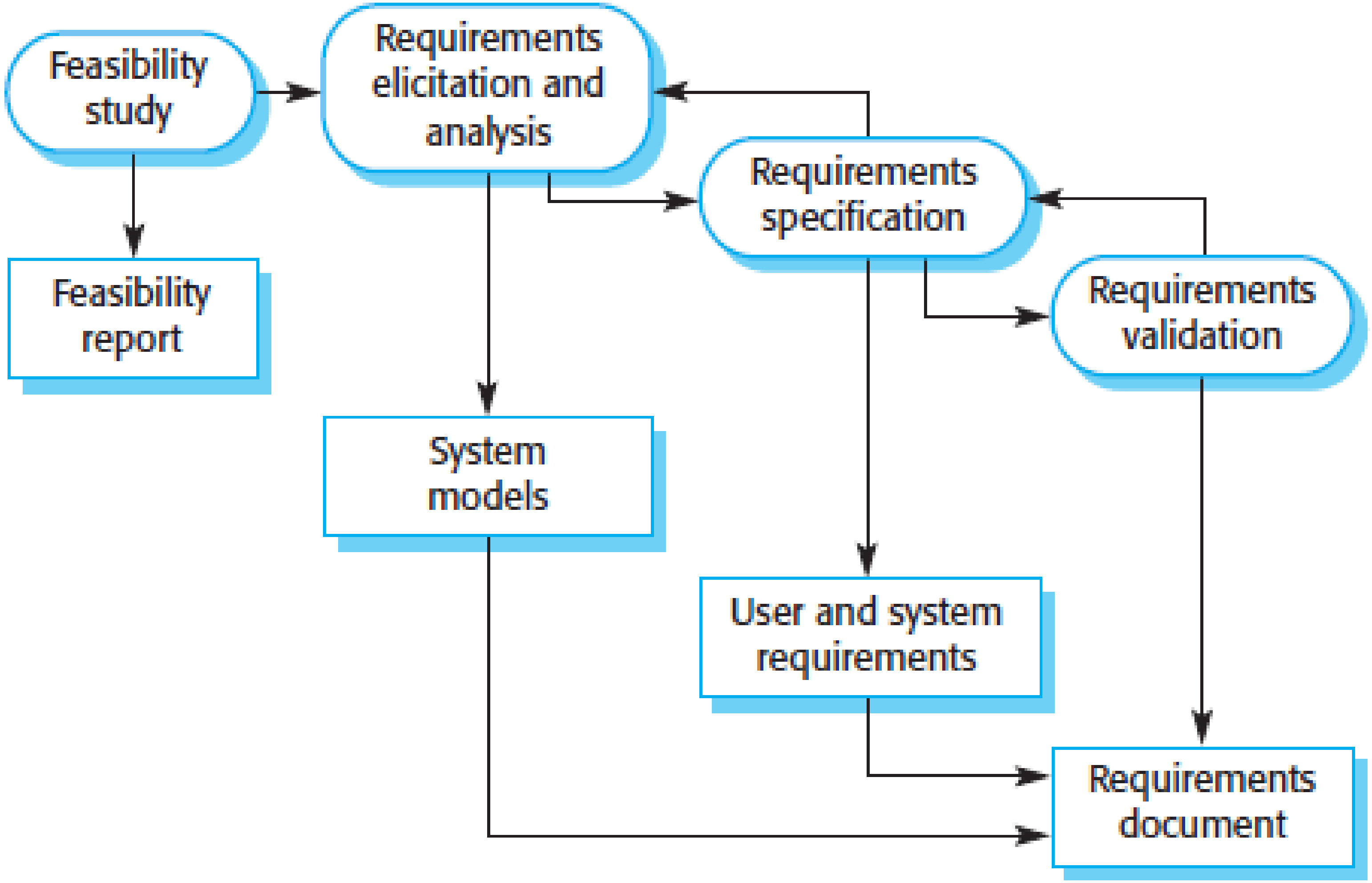
integrate

test

Release

Each cycle:
2 weeks

Software requirement process



Software requirement process (cont.)

- **Feasibility study**
  - ❑ For all new systems.
  - ❑ It is a set of preliminary business requirements.
  - ❑ An outline description of the system and how the system is intended to support business processes.

⇨The results of the feasibility study should be a report that recommends whether or not it is worth to carry out.

Software requirement process (cont.)

- **Requirements elicitation and analysis**
  - Software engineers work with customers and system end-users to find out about the application domain. What services the system should provide, the required performance of the system, hardware constraints, and so on.
  - The process activities are:
    - Requirements discovery
    - Requirements classification and organization
    - Requirements prioritisation and negotiation
    - Requirements documentation

Software requirement process (cont.)

- **Requirements validation**
  - To show that the requirements actually define the system that the customer wants.
  - It is important because errors in a requirements document can lead to extensive rework costs. These documents should be checked:
    - Validity checks
    - Consistency checks: Requirements in the document should not conflict
    - Completeness checks
    - Realism checks
    - Verifiability

Main references

1. Prof Steve Easterbrook, lecture notes, University of Toronto, Canada.
2. CHAOS Report, Standish Group
3. Software Engineering By Ian Sommerville - 8th Edition, Pearson Education, 2007

Assigment

- Chapter 1, Karl Wiegers, Joy Beatty, Software Requirements, Third edition.

Q&A