Cornell University
Computing and Information Science

CS 5150 Software Engineering

6. Requirements Analysis

William Y. Arms

Requirements

Requirements define the function of the system **from the client's viewpoint**.

- The requirements establish the system's functionality, constraints, and goals by consultation with the client, customers, and users.
- The requirements may be developed in a self-contained study, or may emerge incrementally.
- The requirements form the basis for acceptance testing.

The development team and the client need to work together closely during the requirements phase of a software project.

- The requirements must be developed in a manner that is understandable by both the client and the development staff.

Why are Requirements Important?

Causes of failed software projects

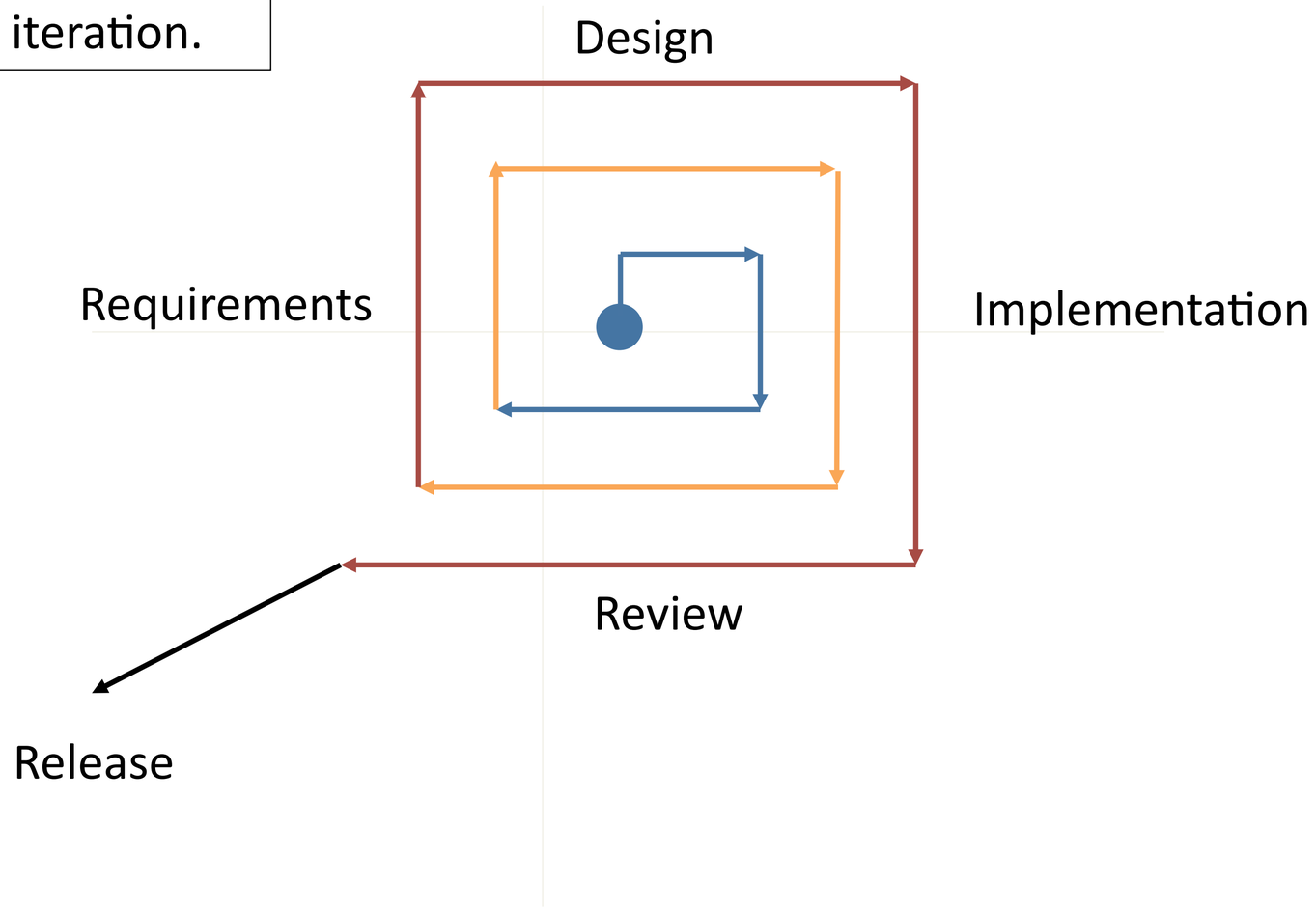
Incomplete requirements	13.1%
Lack of user involvement	12.4%
Lack of resources	10.6%
Unrealistic expectations	9.9%
Lack of executive support	9.3%
Changing requirements & specifications	8.8%
Lack of planning	8.1%
System no longer needed	7.5%

Failures to understand the requirements led the developers to build the wrong system.

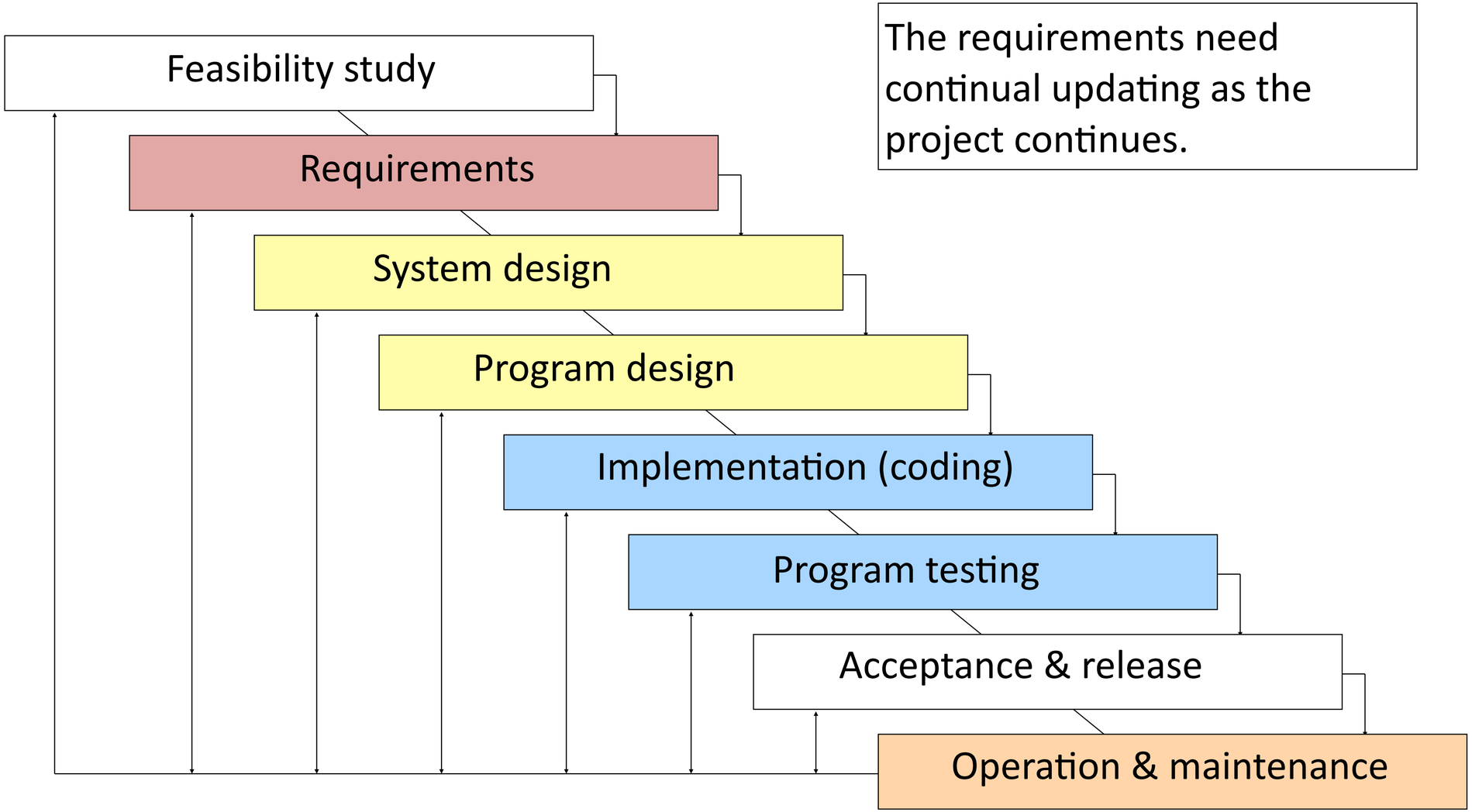
Source: Standish Group

Requirements in Iterative Refinement

The requirements are revised for each iteration.

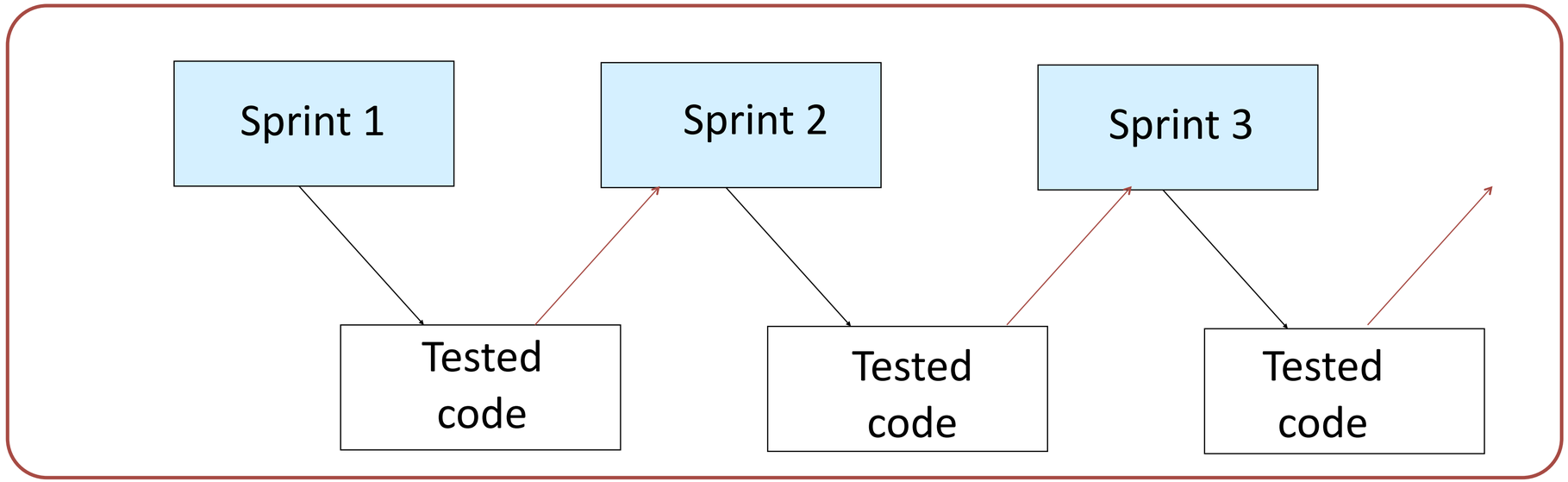


Requirements in the Modified Waterfall Model



Requirements with Agile Development

Each sprint has its own set of requirements.



Requirement Goals

- **Understand** the requirements **in appropriate detail**.
- **Define** the requirements in a manner that is **clear to the client**. This may be a written specification, prototype system, or other form of communication.
- **Define** the requirements in a manner that is **clear to the people** who will **design, implement, and maintain the system.**
- **Ensure** that the **client and developers understand** the requirements and their **implications.**

Many CS 5150 projects use the first presentation and the accompanying report to confirm the requirements with the client.

"Our understanding of your requirements is that ...”

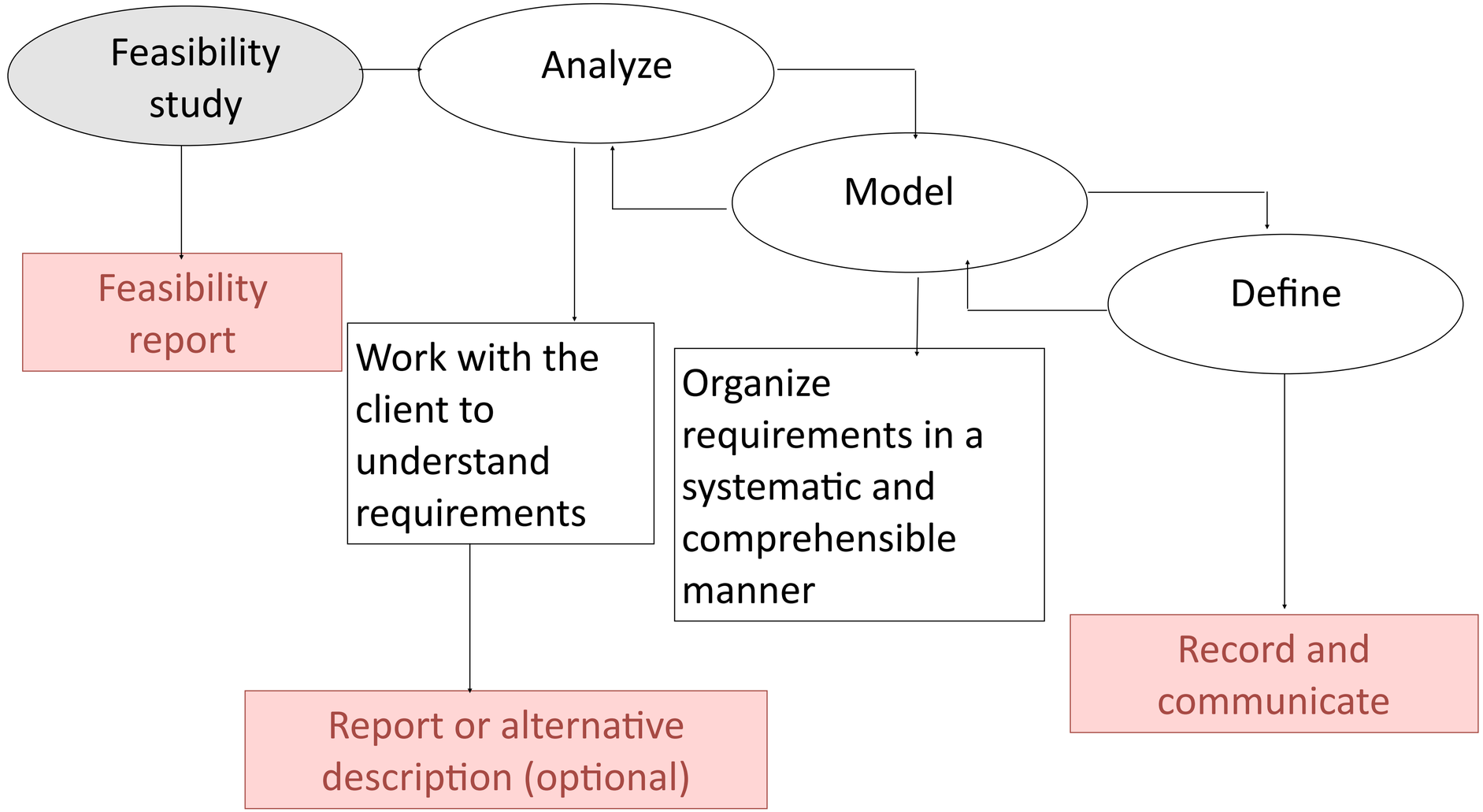
Steps in the Requirements Phase

The requirements part of a project can be divided into several stages:

- **Analysis** to establish the system's services, constraints, and goals by consultation with client, customers, and users.
- **Modeling** to organize the requirements in a systematic and comprehensible manner.
- **Define, record, and communicate** the requirements.

With iterative and agile methods, these stages will be repeated several times.

The Requirements Process



Requirements Analysis: Interviews with Clients

Client interviews are the heart of the requirements analysis.

Clients may have only a vague concept of requirements.

- Allow plenty of time.
- Prepare before you meet with the client.
- Keep full notes.
- If you do not understand, delve further, again and again.
- Repeat what you hear.

For your CS 5150 projects you will need to schedule several meetings with your client to analyze the requirements.

Small group meetings are often most effective.

Requirements Analysis: Understand the Requirements

Understand the requirements in depth

- Domain understanding

  Example: Manufacturing light bulbs

- Understanding of the real requirements of all stakeholders

  Stakeholders may not have clear ideas about what they require, or they may not express requirements clearly.

  They are often too close to the old way of doing things.

- Understanding the terminology

  Clients often use specialized terminology. If you do not understand it, ask for an explanation.

Keep asking questions, *"Why do you do things this way?" "Is this essential?" "What are the alternatives?"*

Requirements Analysis: New and Old Systems

A new system is when there is no existing system. This is rare.

A replacement system is when a system is built to replace an existing system.

A legacy system is an existing system that is not being replaced, but must interface to the new system.

Clients often confuse the current system with the underlying requirement.

In requirements analysis it is important to distinguish:

- features of the current system that are needed in the new system
- features of the current system that are not needed in the new system
- proposed new features

Requirements Analysis: Unspoken Requirements

Discovering the unspoken requirements is often the most difficult part of developing the requirements.

Examples:

- Resistance to change
- Departmental friction (e.g., transfer of staff)
- Management strengths and weaknesses

Requirements Analysis: Stakeholders

Identify the stakeholders:

Who is affected by this system?

- Client
- Senior management
- Production staff
- Computing staff
- Customers
- Users (many categories)
- *etc., etc., etc.,*

Example:

Web shopping site (shoppers, administration, finance, warehouse)

CS 5150 projects that build web applications often find that the administrative system that is not seen by the users is bigger than the part of the site that is visible to the users.

Requirements Analysis: Viewpoint Analysis

Viewpoint Analysis

Analyze the requirements as seen by each group of stakeholders.

Example: University Admissions System

- Applicants
- University administration
  - Admissions office
  - Financial aid office
  - Special offices (e.g., athletics, development)
- Academic departments

- *Computing staff*
- *Operations and maintenance*

Requirements Analysis: Special Studies

Understanding the requirements may need studies:

Market research

focus groups, surveys, competitive analysis, etc.

Example: Windows XP T-shirt that highlighted Apple's strengths

Technical evaluation

experiments, prototypes, etc.

Example: Windows XP boot faster

Defining and Communicating Requirements

Objectives

Record agreement between clients and developers

- Provide a basis for acceptance testing
- Provide visibility
- Be a foundation for system and program design
- Communicate with other teams who may work on or rely on this system
- Inform future maintainers

Defining and Communicating Requirements: Realism and Verifiability

Requirements must be **realistic**, i.e., it must be possible to meet them.

Wrong: *The system must be capable of x (if no known computer system can do x at a reasonable cost).*

Requirements must be **verifiable**, i.e., since the requirements are the basis for acceptance testing, it must be possible to test whether a requirement has been met.

Wrong: *The system must be easy to use.*

Right: *After one day's training an operator should be able to input 50 orders per hour.*

Defining and Communicating Requirements: Option 1 – Heavyweight Processes

Heavyweight software processes expect detailed specification

- Written documentation that specifies each requirement in detail.
- Carefully checked by client and developers.
- May be a contractual document.

Difficulties

- Time consuming and difficult to create.
- Time consuming and difficult to maintain.
- Checking a detailed specification is difficult and tedious.
- Details may obscure the overview of the requirements.
- Clients rarely understand the implications.

The difficulty of creating and maintaining a detailed requirements specification is one of the reasons that many organizations are attracted to lighter weight development processes.

Defining and Communicating Requirements: Option 2 – Lightweight Processes

Lightweight processes use a outline specification + other tools

- Documentation describing key requirements in appropriate detail.
- Reviewed by client and developers.

Details provided by supplementary tools, e.g.,

- User interface mock-up or demonstration.
- Models, e.g., data base schema, state machine, etc.

Clients understand prototypes and models better than specification

- Iterative or incremental (agile) development processes allows the client to appreciate what the final system will do.

Lightweight Processes (continued)

With lightweight processes, experience and judgment are needed to distinguish between details that can be left for later in the development process and key requirements that must be agreed with the client early in the process.

Examples where detailed specifications are usually needed

- Business rules (e.g., reference to an accounting standard)
- Legal restraint (e.g., laws on retention of data, privacy)
- Data flow (e.g., sources of data, data validation)

A common fault is to miss crucial details. This results in misunderstandings between the client and the developers. Yet the whole intent of lightweight processes is to have minimal intermediate documentation.

Requirements in Government Systems

Government systems in the USA and abroad have a reputation for poor functionality combined with delays and cost over-runs.

My personal opinion is that the method of contracting is a major contributor to these problems.

- Responsible use of taxpayers' money leads to contracts in which each sub-process has a defined deliverable at an agreed price.
- In particular most contracts demand a detailed requirement specification before the contract for design and implementation are placed.
- This leads to a waterfall model of development, often with penalties for modifications of the requirements.

But in many government systems the requirements are not well understood.

- They should use a development process in which the requirements are flexible and the design evolves iteratively.
- Contracts for such development processes are difficult to write, but they lead to better software.

Functional Requirements

Functional requirements describe the functions that the system must perform. They are identified by analyzing the use made of the system and include topics such as:

- Functionality
- Data
- User interfaces

Non-Functional Requirements

Requirements that are not directly related to the functions that the system must perform

Product requirements

performance, reliability, portability, etc...

Organizational requirements

delivery, training, standards, etc...

External requirements

legal, interoperability, etc...

Marketing and public relations

Example:

In the NSDL, the client (the National Science Foundation) wanted a live system that could be demonstrated to senior managers six months after the grant was awarded.

Example of Non-Functional Requirements

Example: Library of Congress Repository

Use technology that the client's staff are familiar with:

- Hardware and software systems (IBM/Unix)
- Database systems (Oracle)
- Programming languages (C and C++)

Recognize that the client is a federal library

- Regulations covering government systems, e.g., accessibility to people with disabilities
- Importance of developing a system that would be respected by other major libraries

Requirements: Negotiation with the Client

Some requests from the client may conflict with others:

Recognize and resolve conflicts

(e.g., functionality v. cost v. timeliness)

Example:

Windows XP boot faster: shorter time-out v. recognize all equipment on bus

Requirements: Negotiation with the Client

Sometimes the client will request functionality that is very expensive or impossible. What do you do?

- Talk through the requirement with the client. Why is it wanted? Is there an alternative that is equivalent?
- Explain the reasoning behind your concern. Explain the technical, organizational, and cost implications.
- Be open to suggestions. Is there a gap in your understanding? Perhaps a second opinion might suggest other approaches.

Before the requirements phase is completed the client and development team must resolve these questions.

Example

National Science Foundation grant system, client asked for every format that had ever been used in proposals (e.g., scientific samples). After negotiation, agreed that the first phase would support only PDF.

Requirements Analysis v. System Design

Dilemma about technical decisions

- Requirements analysis should make minimal assumptions about the system design.
- But the requirements definition must be consistent with computing technology and the resources available.

In practice, analysis and design are interwoven. However:

1. *Do not allow assumptions about the design to influence the requirements analysis.*
2. *Do not to allow the requirements analysis to prejudge the system design.*