

# IF3141

## SISTEM INFORMASI

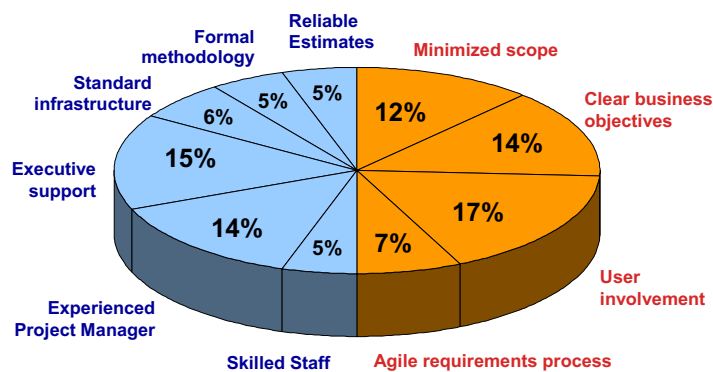
### Requirement Engineering

Semester II 2024/2025

WISATA TERBUKA INFORMATIKA ITS

1

## Project Success Factors

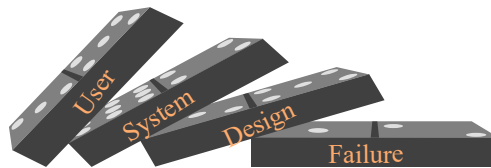


Source: "Chaos Chronicles, III, 2003". [www.standishgroup.com](http://www.standishgroup.com)

2

## Efek Domino

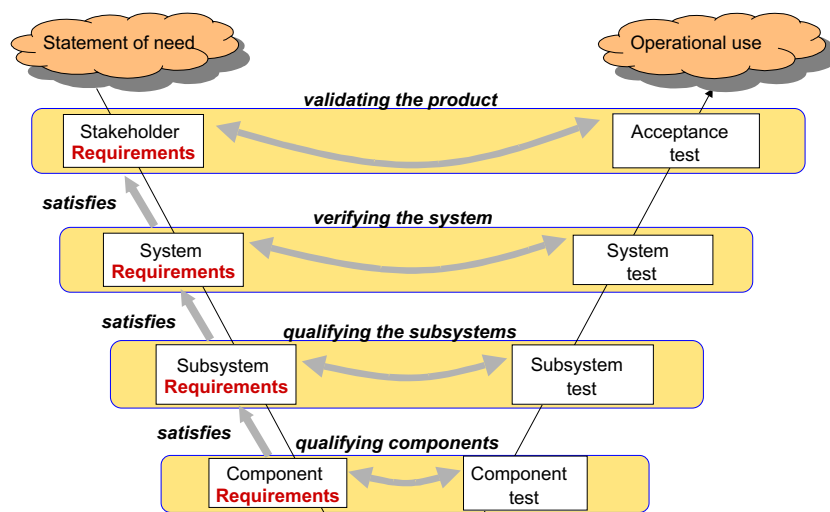
- Ketidak tepatan kebutuhan mempunyai efek domino terhadap siklus hidup
- Kekeliruan kebutuhan pemakai menjurus kekeliruan terhadap kebutuhan sistem yang berakibat kekeliruan rancangan elemen, kekeliruan fungsi dan akhirnya kegagalan



- Berkisar 60% - 70% kegagalan Proyek IT disebabkan karena lemahnya penentuan kebutuhan, analisis dan manajemen .  
- Meta Group, March 2003

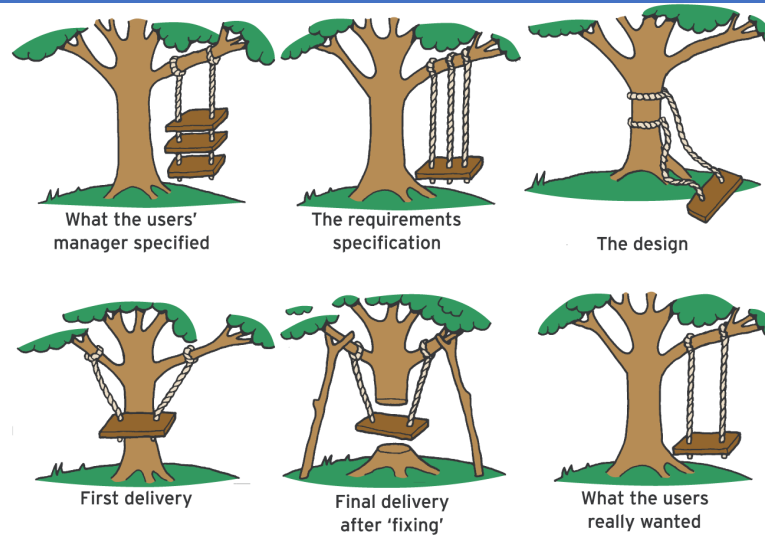
3

## Different levels of Requirements



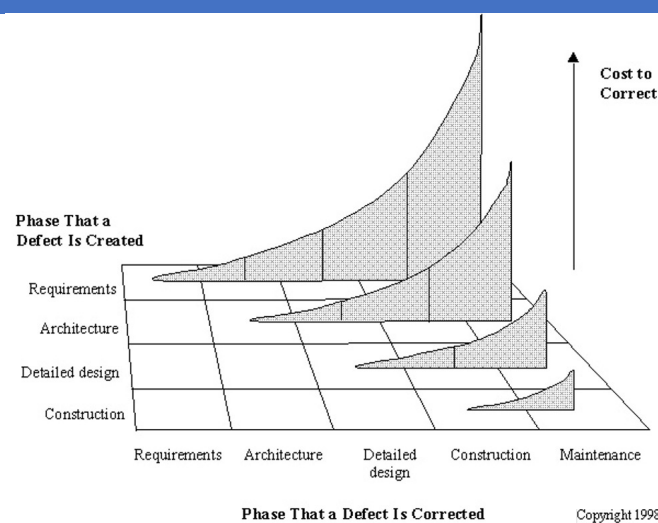
4

## Variasi Interpretasi atas kebutuhan user



5

## Requirements Engineering is Important



Copyright 1998 Steven C. McConnell. Reprinted with permission from *Software Project Survival Guide* (Microsoft Press, 1998).

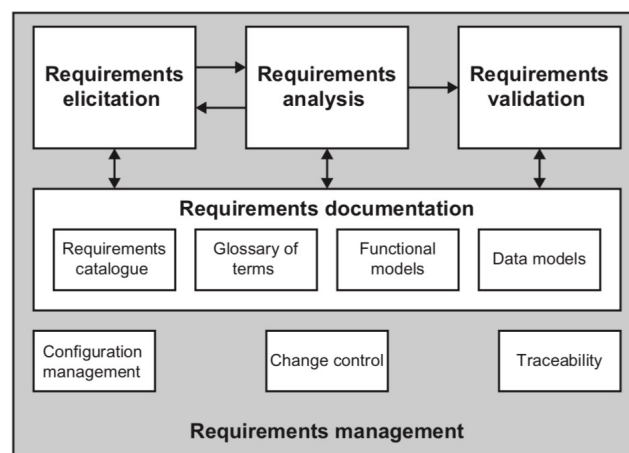
6

## Cost of Errors

Phase in Which Error Discovered	Cost Ratio
Requirements	1
Design	3-6
Coding	10
Development Testing	15-40
Acceptance Testing	30-70
Operation	40-1000

7

## A framework for Requirement Engineering



8

## A framework for Requirement Engineering

- **Requirements Elicitation**

To discover what the requirements are and the best term for this process of discovery is elicitation.

Many techniques that can be employed at this stage, depending on the circumstances of the individual project.

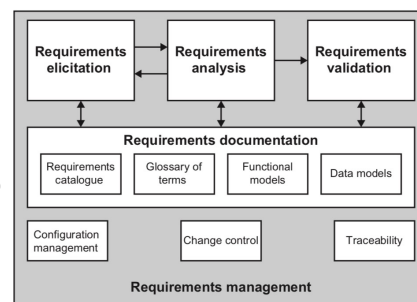
- **Requirements Analysis**

The 'raw' requirements are now examined methodically to see if they meet the standards of 'good' requirements.

Further elicitation work is often required to ensure that they meet these criteria.

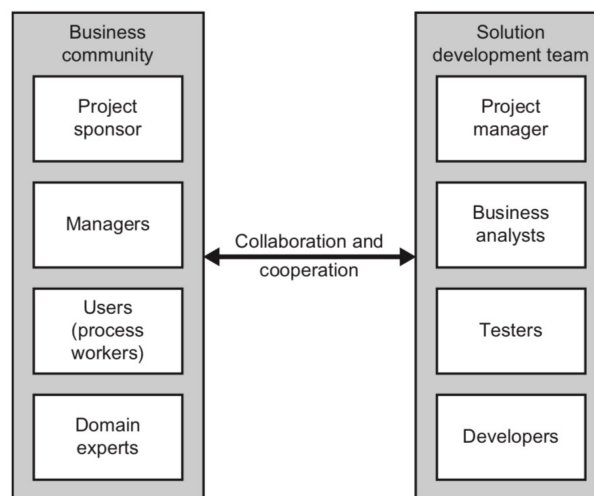
- **Requirements Validation**

The requirements are reviewed and signed off before being used for the next stage.



9

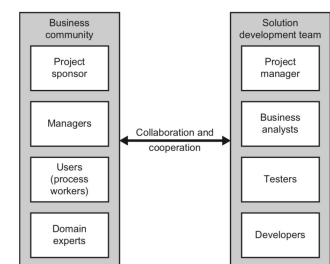
## Roles in Requirements Engineering



10

## Roles in Requirements Engineering

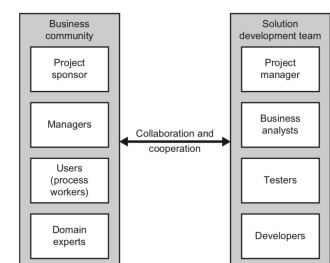
- **Project Sponsor**  
Own the project and is *accountable for delivery* of its benefit  
*Makes the final decision* on key aspects of the project
- **Managers**  
Have *inputs to the requirements process*, as the 'owners' of individual requirements
- **Users (Process worker)**  
Work with the existing systems and processes (if these exist) and who *will have to use the new solution* are a key source of information in requirements engineering.
- **Domain Experts**  
People within the organisation with a *wider view* (perhaps because they have previously worked in other similar organisations); or, domain consultants that can be engaged to give an alternative view.



11

## Roles in Requirements Engineering

- **Project Manager**  
Has *overall charge of the development project* and *needs to keep control* of the scope of the project
- **Business Analysts**  
Responsible for *eliciting, documenting, analysing and managing* the requirements.
- **Testers**  
Who *review the requirements* (during analysis and validation) to ensure that these are, ultimately, testable.
- **Developers**  
Need to have *good, clear requirements from which to work*, so it is useful to consult them during the requirements engineering work. In Agile approach, the developers will be responsible for fleshing out much of the detail of the requirements as they work with the users to create the solution.



12

## Requirements Elicitation

- *Discovering, uncovering, and understanding* what stakeholders need, want, and expect from the future system.
  - It's not just asking, "What do you want the system to do?" — but it's about **drawing out the real needs**, even when users may not know how to express them clearly.
- (It is not just about **asking**—it's about **listening, interpreting, and validating, but** to uncover the "why" behind each request.)

13

## Requirements Elicitation

### Objectives of Requirement Elicitation

- Understand the **business goals** and **pain points**.
- Identify and engage **all relevant stakeholders**.
- Document both **functional** and **non-functional** requirements.
- Surface any **conflicts** or **constraints** early.

14

## Requirements Elicitation

### Challenges in Requirement Elicitation

Challenge	Description
<b>Unstated Needs</b>	Users assume things that they don't articulate.
<b>Stakeholder Misalignment</b>	Conflicting needs from different departments.
<b>Limited Technical Vocabulary</b>	Users may not know how to describe what they need.
<b>Changing Requirements</b>	Needs evolve as understanding increases.
<b>Cognitive Biases</b>	Stakeholders may prioritize recent issues or familiar solutions.

15

## Requirements Elicitation

- **The difficulties of Requirement Elicitation**
  - Needs vs Wants
  - Knowing or Thinking that they know or do not know that they know
- **Techniques**
  - Interviews
  - Workshops
  - Focus Groups
  - Observation
  - Shadowing
  - Scenarios
  - Prototyping

16



## Requirements Elicitation

- **Steps in the Elicitation Process**
  - A **cycle** of learning and refining:
- 1. **Identify Stakeholders**
  - Who uses the system? Who benefits from it? Who maintains it?
- 2. **Select Elicitation Techniques**
  - Interviews, workshops, focus groups, observation, etc.
- 3. **Prepare for Elicitation**
  - Understand the domain, prepare questions, set objectives for each session.
- 4. **Conduct Elicitation**
  - Use the selected methods to gather information. Capture **verbal and non-verbal cues**, decisions, hesitations.
- 5. **Document Requirements**
  - Write clear, structured requirements. Avoid technical jargon when possible.
- 6. **Confirm and Refine**
  - Go back to stakeholders to **validate interpretations**, check for completeness.

17

## Differences of Requirements Elicitation Techniques

### 1. Interviews

- One-on-one conversations with stakeholders to understand their needs.
- For in-depth understanding of individual perspectives.
- **Pros:** Personal, can clarify answers, allows deep probing.
- **Cons:** Time-consuming, may miss group dynamics.

### 2. Workshops

- Collaborative sessions involving multiple stakeholders to discuss and define requirements together.
- For resolving conflicting needs, building consensus.
- **Pros:** Fast requirement gathering, good for prioritization.
- **Cons:** Needs careful facilitation, not all voices may be heard.

### 3. Focus Groups

- Group discussions with selected stakeholders (often users) guided by a moderator.
- For exploring opinions, user attitudes, and expectations.
- **Pros:** Good for gathering a variety of views, discovering new ideas.
- **Cons:** Groupthink, dominant personalities may overshadow others.

### 4. Observation

- Watching users perform their tasks in their natural environment.
- Understanding actual workflows and pain points.
- **Pros:** Reveals tacit knowledge and real behaviors.
- **Cons:** May be intrusive, time-consuming, users might act differently when observed.

18

## Requirements Elicitation Techniques

### 5. Shadowing

- A form of observation where the analyst follows a user closely throughout their workday.
- For deep insight into complex or expert workflows.
- **Pros:** High-detail understanding of tasks.
- **Cons:** Resource-intensive, users may feel uncomfortable.

### 6. Scenarios

- Narrative descriptions of how users interact with the system in a given situation.
- For exploring “what-if” cases and validating requirements.
- **Pros:** Helps imagine future system use, aligns expectations.
- **Cons:** Hypothetical—may not reflect actual behavior.

### 3. Prototyping

- Creating a simple mock-up or working model of the system for users to interact with.
- For getting feedback on system design and usability.
- **Pros:** Visual and interactive, reduces misunderstanding.
- **Cons:** Can set false expectations if not properly managed.

19

## Requirements Elicitation Techniques

Technique	Interaction	Depth	Suitable For	Risk
Interviews	1-on-1	High	Detailed needs	Slow process
Workshops	Group	High	Conflicting views, prioritization	Needs facilitation
Focus Groups	Group	Medium	Exploring perceptions	Bias risk
Observation	Passive	High	Real behaviors	Observer effect
Shadowing	Passive	Very High	Complex tasks	Time-intensive
Scenarios	Hypothetical	Medium	Exploring future use	Less realistic
Prototyping	Visual	High	Design feedback	Unrealistic expectations

20

## Types of Requirements

- **Functional** (perilaku)
  - What a proposed solution is expected to do, in other words its functionality.
  - An example might be 'The system shall identify when an invoice is overdue for payment and alert the financial controller.'
- **Non-functional (a.k.a. "technical")**
  - How the solution should operate
    - Usability: Human factors, help, documentation
    - Reliability: Failure rates, recoverability, availability
    - Performance: Response times, throughput, resource usage
    - Supportability: Maintainability, internationalization
    - Operations: systems management, installation
    - Interface: integration with other systems
    - Other: legal, packaging, hardware

21

## PIECES Classification of System Requirements

Nonfunctional Requirement Type	Explanation
Performance	Performance requirements represent the performance the system is required to exhibit to meet the needs of users. <ul style="list-style-type: none"> <li>• What is the acceptable throughput rate?</li> <li>• What is the acceptable response time?</li> </ul>
Information	Information requirements represent the information that is pertinent to the users in terms of content, timeliness, accuracy, and format. <ul style="list-style-type: none"> <li>• What are the necessary inputs and outputs? When must they happen?</li> <li>• What is the required data to be stored?</li> <li>• How current must the information be?</li> <li>• What are the interfaces to external systems?</li> </ul>
Economy	Economy requirements represent the need for the system to reduce costs or increase profits. <ul style="list-style-type: none"> <li>• What are the areas of the system where costs must be reduced?</li> <li>• How much should cost be reduced or profits be increased?</li> <li>• What are the budgetary limits?</li> <li>• What is the timetable for development?</li> </ul>
Control (and security)	Control requirements represent the environment in which the system must operate, as well as the type and degree of security that must be provided. <ul style="list-style-type: none"> <li>• Must access to the system or information be controlled?</li> <li>• What are the privacy requirements?</li> <li>• Does the criticality of the data necessitate the need for special handling (backups, off-site storage, etc.) of the data?</li> </ul>
Efficiency	Efficiency requirements represent the system's ability to produce outputs with minimal waste. <ul style="list-style-type: none"> <li>• Are there duplicate steps in the process that must be eliminated?</li> <li>• Are there ways to reduce waste in the way the system uses its resources?</li> </ul>
Service	Service requirements represent needs in order for the system to be reliable, flexible, and expandable. <ul style="list-style-type: none"> <li>• Who will use the system, and where are they located?</li> <li>• Will there be different types of users?</li> <li>• What are the appropriate human factors?</li> <li>• What training devices and training materials are to be included in the system?</li> <li>• What training devices and training materials are to be developed and maintained separately from the system, such as stand-alone computer-based training (CBT) programs or databases?</li> <li>• What are the reliability/availability requirements?</li> <li>• How should the system be packaged and distributed?</li> <li>• What documentation is required?</li> </ul>

22

## PIECES Classification of System Requirements

### The PIECES Problem-Solving Framework and Checklist

The following checklist for problem, opportunity, and directive identification uses Wetherbe's PIECES framework. Note that the categories of PIECES are not mutually exclusive; some possible problems show up in multiple lists. Also, the list of possible problems is not exhaustive. The PIECES framework is equally suited to analyzing both manual and computerized systems and applications.

#### PERFORMANCE

- A. Throughput – the amount of work performed over some period of time.
- B. Response times – the average delay between a transaction or request, and a response to that transaction or request.

#### INFORMATION (and Data)

- A. Outputs
  1. Lack of any information
  2. Lack of necessary information
  3. Lack of relevant information
  4. Too much information – “information overload”
  5. Information that is not in a useful format
  6. Information that is not accurate
  7. Information that is difficult to produce
  8. Information is not timely to its subsequent use
- B. Inputs
  1. Data is not captured
  2. Data is not captured in time to be useful
  3. Data is not accurately captured – contains errors
  4. Data is difficult to capture
  5. Data is captured redundantly – same data captured more than once
  6. Too much data is captured
  7. Illegal data is captured
- C. Stored data
  1. Data is stored redundantly in multiple files and/or databases
  2. Same data items have different values in different files (poor data integration)
  3. Stored data is not accurate
  4. Data is not secure to accident or vandalism
  5. Data is not well organized
  6. Data is not flexible – not easy to meet new information needs from stored data
  7. Data is not accessible

#### ECONOMICS

- A. Costs
  1. Costs are unknown
  2. Costs are untraceable to source
  3. Costs are too high
- B. Profits

1. New markets can be explored
2. Current marketing can be improved
3. Orders can be increased

#### CONTROL (and Security)

- A. Too little security or control
  1. Input data is not adequately edited
  2. Crimes (e.g., fraud, embezzlement) are (or can be) committed against data
  3. Ethics are breached on data or information – refers to data or information getting to unauthorized people
  4. Redundantly stored data is inconsistent in different files or databases
  5. Data privacy regulations or guidelines are being (or can be) violated
  6. Processing errors are occurring (either by people, machines, or software)
  7. Decision-making errors are occurring
- B. Too much control or security
  1. Bureaucratic red tape slows the system
  2. Controls inconvenience customers or employees
  3. Excessive controls cause processing delays

#### EFFICIENCY

- A. People, machines, or computers waste time
  1. Data is redundantly input or copied
  2. Data is redundantly processed
  3. Information is redundantly generated
- B. People, machines, or computers waste materials and supplies
- C. Effort required for tasks is excessive
- D. Material required for tasks is excessive

#### SERVICE

- A. The system produces inaccurate results
- B. The system produces inconsistent results
- C. The system produces unreliable results
- D. The system is not easy to learn
- E. The system is not easy to use
- F. The system is awkward to use
- G. The system is inflexible to new or exceptional situations
- H. The system is inflexible to change
- I. The system is incompatible with other systems

23

## Requirements Analysis

- The process of **examining, refining, structuring, and prioritizing** the gathered requirements to ensure they are **complete, clear, and actionable** for system design and development.
- It transforms “raw” elicited needs into **high-quality requirements** that can guide system building.

24

## Requirements Analysis

- **Objectives of Requirement Analysis**
  - Ensure **clarity** and **consistency** in all requirements.
  - Detect and **resolve conflicts** between requirements.
  - Prioritize requirements based on value and feasibility.
  - Ensure all requirements are **testable**, **achievable**, and **traceable**.
  - Prevent **costly errors** later in the development process.

25

## Requirements Analysis

- **Key Activities in Requirement Analysis**
  - 1. Clarification & Refinement**
    - Clean up vague language (e.g., replace “fast” with measurable benchmarks).
    - Ask “what do you mean by...” questions to refine requirements.
  - 2. Categorization**
    - Group requirements into:
      - **Functional** – system behaviors
      - **Non-functional** – system qualities (performance, security, usability)

Use PIECES framework (Performance, Information, Economy, Control, Efficiency, Service)

26

## Requirements Analysis

### 3. Validation of Good Requirement Characteristics

Use a checklist like:

Quality	Description
<b>Correct</b>	Accurately reflects user needs
<b>Unambiguous</b>	Only one possible interpretation
<b>Testable</b>	Can be verified by testing
<b>Feasible</b>	Technically and financially possible
<b>Traceable</b>	Can trace back to stakeholder
<b>Atomic</b>	Represents one single requirement
<b>Consistent</b>	No conflict with other requirements

Example:

- ✗ "System should be easy to use"
- ✓ "80% of first-time users should complete a task within 5 minutes without external help"

27

## Requirements Analysis

### 4. Modeling

- Use tools such as:
  - **Use Case Diagrams**
  - **Entity-Relationship Diagrams**
  - **Data Flow Diagrams**
- These help *visualize* and *validate logic, data, and flow*

### 5. Prioritization

- Use **MoSCoW** method:
  - **Must Have**
  - **Should Have**
  - **Could Have**
  - **Won't Have (for now)**

28

## Requirements Analysis: Characteristics of good requirements

- **Categorised**  
Each requirement needs to be categorised to make it easier to group requirements for review.
- **Relevant**  
Is the requirement within the scope of the project?  
If it is outside the scope, the project sponsor must be consulted to see if they want to broaden the scope to include it. If so, then the proper change control process should be employed.
- **Prioritised**  
Not all requirements are the same priority
- **Achievable**  
If it is clear that the requirement cannot be met – technically impossible or too expensive – then the expectations of whoever raised it need to be managed downwards and, ultimately, the requirement needs to be removed from the requirements document.

29

## Requirements Analysis: Characteristics of good requirements

- **Understandable and unambiguous**  
Do the users, for example, have the same understanding of what a 'customer' is as the solution developers?  
For example, 'a fast response time' is ambiguous, 'a response time of not more than two seconds after hitting the ENTER key' is not.
- **Testable**  
Requirements have to be testable.  
Concepts such as 'ease of use' should be specified in such a way as to make them testable; for example, by defining how long it should take a typical user to master a feature, and with what training and support.
- **Requirement not solution**  
It is easy to wrongly document solutions as requirements. For instance, 'the field sales force shall be provided with laptop computers' would be better expressed as 'a means is needed for the field sales force to check product information and record orders when on the move', which would open up other solutions such as the use of tablet computers or smartphones.

30

## Requirements Analysis: Characteristics of good requirements

- **Consistent**  
Should not be conflict with each others  
For example, 'the system must be totally secure' is inconsistent with 'the system must be readily accessible to anyone'.
- **Owned**  
Someone needs to own each requirement.  
The owner of a requirement is likely to be someone in the management tier who can make decisions about the relevance and content of that requirement.
- **Unique and atomic**  
Each requirement is singular and doesn't cover several requirements.
- **Traceable**  
Where a requirement came from should be clear  
In the end, what happened to the requirement should be known. Was it met in the delivered solution or perhaps set aside as too expensive?

31

## Requirements Analysis: Characteristics of good requirements

- **Concise**  
Requirements as brief as possible, as this makes them easier to understand and review.
- **Complete**  
It must be a complete statement of what is required and not necessitate extensive reference to other documentation.
- **Correct**  
Each requirement should be correct.  
However, a requirement could still be excellent in a technical sense and yet plain wrong in the sense of wrongly defining what the organisation needs.
- **Conformant**  
If the organisation has a set of standards as to how requirements should be documented and phrased, these should be followed.

32



## Requirements Analysis: Prioritization of Requirements

- **Must Have**  
Fundamental or mandatory requirements that, if they are not satisfied, would render any solution unacceptable or useless.  
For example, the main point of an inventory management system is to record items taken into and out of a storage facility so, without the ability to do that, any system would be of no value.
- **Should Have**  
Mandatory but, unlike the 'Must haves', they do not necessarily have to be met on day one.  
Either implementation can be delayed, or the users can continue to use their old system or a manual workaround may be acceptable for a short time.
- **Could Have**  
Nice to have. If it is easily implemented with "Must and Should have" requirements they will be implemented, otherwise can be left out of the solution.
- **Want to have (but not now)**

33

## Requirements Validation

- The process of **ensuring the documented requirements are correct, complete, consistent, and approved by the stakeholders.**
- It checks whether the right product is being built—before it's actually built.

 While analysis ensures **requirements are well-written**, validation ensures they're the **right requirements**.

34

## Requirements Validation

- **Verification**  
Here we are concerned with requirements **meeting the standards** set for them
- **Validation**  
It is the responsibility of the users (process workers), their managers and, ultimately, the project sponsor to satisfy themselves that the set of requirements **does fully and correctly encapsulate their needs**.  
If the set of requirements is very large, it may be advisable to partition it so that individual stakeholders are only required to review the parts that are most relevant to them, although this does risk no-one being able to see the overall picture.

35

## Requirements Validation

- **Objectives of Requirement Validation**
  - Confirm the system will meet **stakeholders' real needs**.
  - Gain formal **approval** from users, managers, and sponsors.
  - Catch and correct mistakes early before costly development begins.
  - Ensure that each requirement is **understood, testable, and accepted** by all relevant parties.


36

## Requirements Validation

- **Key Activities in Requirement Validation**

- 1. **Requirement Review Sessions**

- Conduct **walkthroughs or inspections** involving stakeholders, analysts, developers, and testers.
    - Review:
      - Clarity
      - Testability
      - Alignment with business goals

 **Tip:** Partition requirements by user roles to avoid overwhelming reviewers.

37

## Requirements Validation

- 2. **Validation Criteria Checklist**

Ask the following:

- ✓ Is this what the user actually needs?
- ✓ Is it testable (measurable, verifiable)?
- ✓ Is it complete? No missing scenarios?
- ✓ Is it consistent with other requirements?
- ✓ Has it been agreed upon by all key stakeholders?

38

## Requirements Validation

### 3. Prototyping (Optional)

- Show **low-fidelity mockups** or click-through prototypes to stakeholders to validate that the flow and functionalities make sense.

### 4. Acceptance Criteria Definition

- Define what “done” looks like.
- E.g., “A user can successfully reset their password via email within 5 minutes of request.”

39

## Requirements Validation

### Common Issues Found During Validation

Issue	Result
<b>Missing Requirements</b>	Critical features not implemented
<b>Ambiguous Wording</b>	Different interpretations across teams
<b>Conflicting Requirements</b>	System behaves inconsistently
<b>Unverifiable Requirements</b>	Cannot write proper test cases
<b>Scope Creep</b>	Uncontrolled changes to original requirements

40

## Requirements Validation

- **Outputs of Requirement Validation**
  - **Signed-off Requirement Specification** (SRS or BRD)
  - **Requirements traceability matrix** (RTM)
  - Record of **feedback and revisions**
  - List of **approved and rejected** requirements
- **Who Participates?**
  - Users/Process Workers – Validate actual use
  - Managers – Align with goals & budget
  - Project Sponsor – Approves scope
  - Testers – Ensure testability
  - Business Analysts – Lead the validation process

Validation is your last chance to **prevent misunderstandings** before development starts.  
Don't assume silence = agreement—**get confirmation in writing.**

41

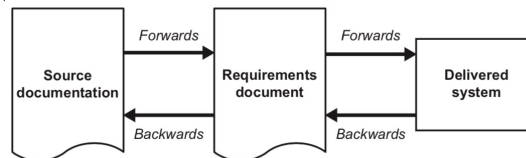
## Requirements Documentation

- **Requirements Catalogue**  
A repository for all requirements
- **Glossary of Terms**  
A way to removing ambiguity from the requirements document.  
If there are multiple definitions of what 'customer' means within an organisation, the glossary makes clear what it means in the context of this document and this project.
- **Functional Models**
  - Data flow diagram or use case diagrams to complement the functional requirements
- **Data Models**
  - To capture the business rules that underpin the storage, use and deletion of data and this can be achieved by using an entity relationship diagram or a class model.

42

## Requirements Management

- **Configuration Management (Version Control)**  
Current status of a requirement, which is the latest version and how all the requirements and other documentation fit together.
- **Change Controls**
  - A process is also needed to document proposed changes to the requirements, their estimated impact on the scope, cost and timescale of the project and what decisions are made about these changes.
- **Traceability**
  - It is important to be able to trace requirements back to their source and forward to their resolution



43

## SUMMARY: What You Should Remember

- Importance of good requirements
- Techniques for gathering them
- Characteristics of good requirements
- How to prioritize and validate

44