Software Design (20VT-2DV608)

Requirements Engineering

Requirements Validation and Management

Francis Palma

francis.palma@lnu.se

Department of Computer Science and Media Technology

Linnaeus University

Course Outline

- Why Software Engineering - Design (Mauro Caporuscio)
  - w4
- Requirement Engineering (Francis Palma)
  - w5.1 (Jan 29th): Understanding and Elicitation of Software Requirements
  - **w5.2 (Jan 31st): Requirements Validation and Management**
  - w6.1 (Feb 5th): Modeling with UML
  - w6.2 (Feb 7th): Requirements Modelling and Management with Tools
  - W7: ASSIGNMENT RE
- Performance Engineering (Diego Perez)
  - w8.1 to w10
- Design and Refactoring (Mauro Caporuscio)
  - w11.1 to w13

Requirements Validation and Management

- Requirements Validation
- Requirements Management

Requirements Validation

- The objectives of requirements validation are to check the set of identified requirements and to discover possible problems with the requirements.
- The process should involve system stakeholders, requirements engineers, and system designers.
- Common requirements problems include:
  - Lack of conformance to quality standards
  - Poorly worded requirements which are ambiguous
  - Requirements conflicts not detected during the analysis process
- These problems must be solved before the requirements document is approved.
  - Iterate over requirements elicitation, analysis and negotiation.

Requirements Analysis *vs.* Requirements Validation

- Both requirements analysis and requirements validation involve finding omissions and conflicts in the system requirements.

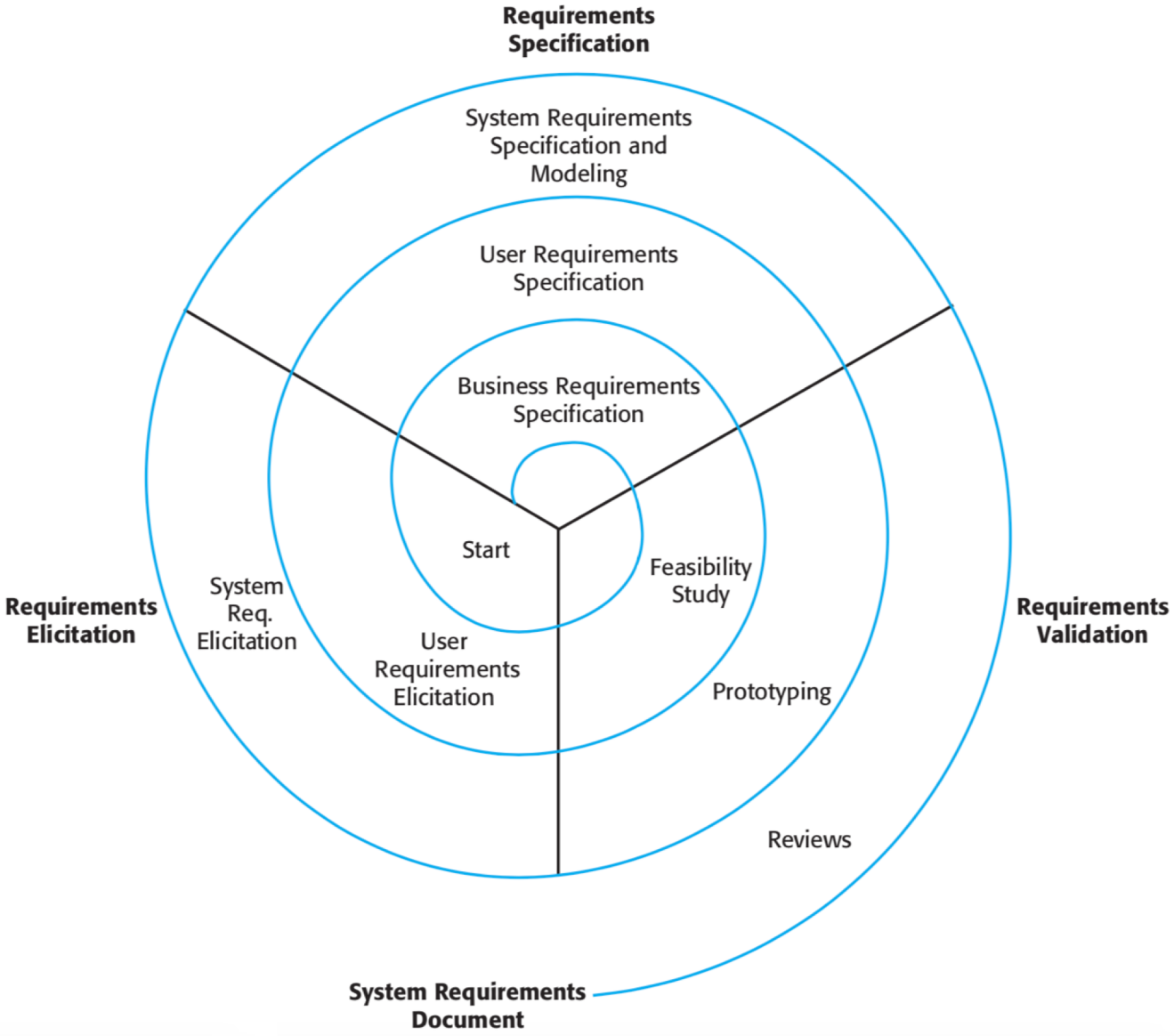
- **So, what is the difference between requirements analysis and requirements validation?**

Requirements analysis is concerned with '**raw**' requirements as elicited from system stakeholders. The requirements are usually incomplete and are expressed in an informal and unstructured way. Notations may or may not be used to describe the requirements.

Requirements validation is concerned with checking a '**final draft**' of a requirements document which includes all system requirements and where known incompleteness and inconsistency has been removed. The document and the requirements should follow defined quality standards.

- Studies showed errors (consequences of requirements errors) in delivered software systems may cost up to 100 times to repair as *programming errors*.

Spiral View of Requirements Engineering Process



Ensure Requirements Document Meets Standards

- Before distributing the requirements document for *general review*, one person should carry out a quick standards check to ensure that the document structure and the requirements are consistent with the standards.
- **Benefit**: Checking against a standard can quickly reveal problems with the requirements specification.
- **Who should perform this?**
  - An analyst or an engineer who is familiar with the requirements standards but *not involved* in the requirements specification.
- **How to perform?**
  - The checker should compare the structure of the requirement document to the standard and highlight missing or incomplete sections.
  - Check pages are numbered, diagrams and figures are labelled, no unfinished or '*to be completed*' requirements.

Ensure Requirements Document Meets Standards

- **What's next, if deviations are found?**
  1) Return the document to the requirements engineering team to correct deviations, if there is enough time to re-issue of the document.
  2) Take note of the deviations from the standards and distribute this to document reviewers.
     - Save the time and cost of creating a new version of the requirements document.
- Choose option 2, when tight deadline for the requirements review or when the deviations from the standard are minor, -- not affect the understandability of the document.

Formal Requirements Inspection

- The system requirements should be validated by a group of people from different backgrounds who systematically check the requirements, meet to discuss problems with the requirements, and agree on how these problems should be fixed.
- **Benefits include:**
  - Formal inspections are a cost-effective way of discovering problems in requirements documents, user documentation, software designs, and programs.
- The requirements inspection is a formal meeting, should be led by someone who has not been involved in producing the requirements.
- During the meeting, a requirements engineer presents each requirement for comment by the group, and identified problems are recorded for discussion.

Actions to Consider for the Identified Problems

Requirements clarification	The requirement may be badly expressed or omitted information collected during requirements elicitation. The author must improve the requirement by rewriting it.
Missing information	Requirements engineers who are revising the document discover this information from system stakeholders or other requirements sources.
Requirements conflict	The involved stakeholders must negotiate to resolve the conflict.
Unrealistic requirement	Stakeholders must be consulted to decide whether the requirement should be deleted or modified to make it more realistic.

Multi-disciplinary Teams to Review Requirements

- Requirements document should be reviewed by a multi-disciplinary team with diverse backgrounds.
  - A system end-user or end-user representative
  - A customer representative
  - One or more domain experts
  - One or more engineers who will be responsible for system design/implementation
  - One or more requirements engineers
- **Benefits include:**
  - People from different backgrounds bring different skills, knowledge, and experience to the review.
  - If system stakeholders from different backgrounds are involved in the review process, they can develop an understanding of the needs of other stakeholders.

Define a Requirements Validation Checklist

- Define a checklist to focus the attention of requirements validators on critical attributes of the requirements document, which identify what readers should look for.
- Benefits include:
  - Checklist adds structure to the validation process. Thus, readers will not forget to check some aspects of the requirements document.
  - Particularly important for managements and end-users who may not have requirements validation experience.

Questions in the Validation Checklist

- Are the requirements complete – does the checker know of any missing requirements or is there any information missing from individual requirement descriptions?
- Are the requirements consistent – do the descriptions of different requirements include contradictions?
- Are the requirements comprehensible – can readers of the document understand what the requirements mean?
- Are the requirements ambiguous – are there different possible interpretations of the requirements?
- Is the requirements document structured – are the descriptions of requirements organized so that related requirements are grouped?
- Are the requirements traceable – do the requirements include links to related requirements and to the reasons why these requirements have been included?
- Does the requirements document as a whole, or do the individual requirements conform to defined standards?

Use Prototyping

- Implement a prototype of the system for experimentation. Stakeholders can try the system to see if it meets their real needs and can suggest improvements.

- Benefits include:
  1. It is difficult to visualize a written statement of requirements. A prototype to demonstrate poorly understood requirements helps stakeholders to discover problems and suggest improvement.
  2. The prototype can be used to develop system tests.
  3. An executable prototype may serve as a stop-gap system which can be delivered if there are delays in implementing the final system.
  4. Prototype implementation requires careful study of the requirements, which may reveal requirements inconsistencies and incompleteness.

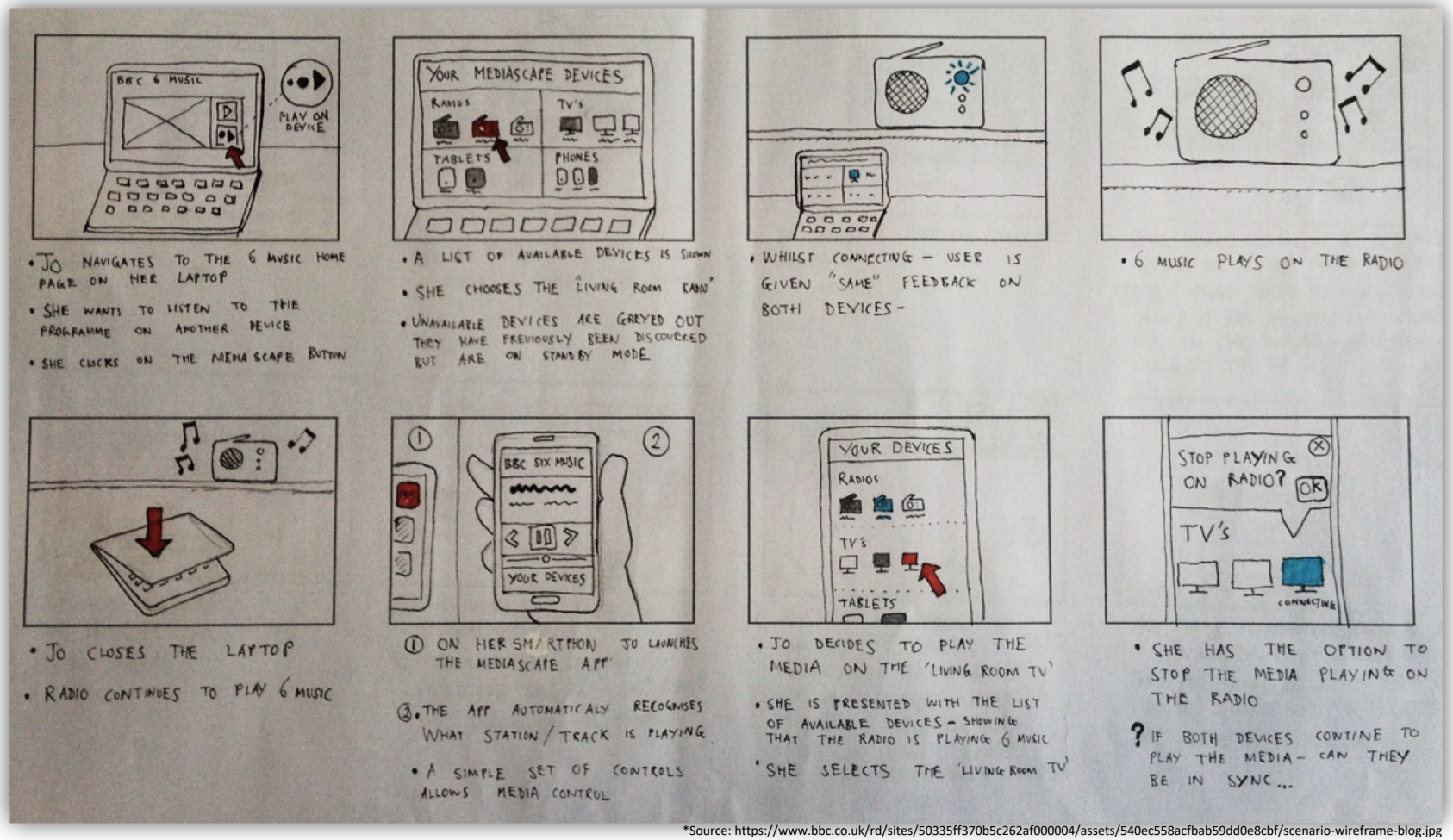
- *For some types of system, the final system can be developed by modifying and adding facilities to the prototype.*

Possible Approaches to Prototyping: Paper Prototyping

- Paper prototyping, a mock-up of the system is developed and used for system experiments.
  - Paper prototyping is a cheap and effective approach to prototype development. It is most suitable for establishing end-user requirements for software systems.
  - Paper versions of the screens (presented to the end-user) are drawn and various usage scenarios are planned.
  - Analysts and end-users' work through the scenarios to find users reactions to the system, the information they require, and how they would normally interact with the system.

Possible Approaches to Prototyping: Paper Prototyping

- An example of Paper Prototyping for a requirement:
  - *"Jo wants to listen to a radio program on a radio device rather than her laptop; she finds the available devices, chooses one, and the radio program plays on the radio device."*
- Let's say we could implement this in two ways:
  - by sending the stream to the device via the laptop;
  - by telling the radio device to play the stream.



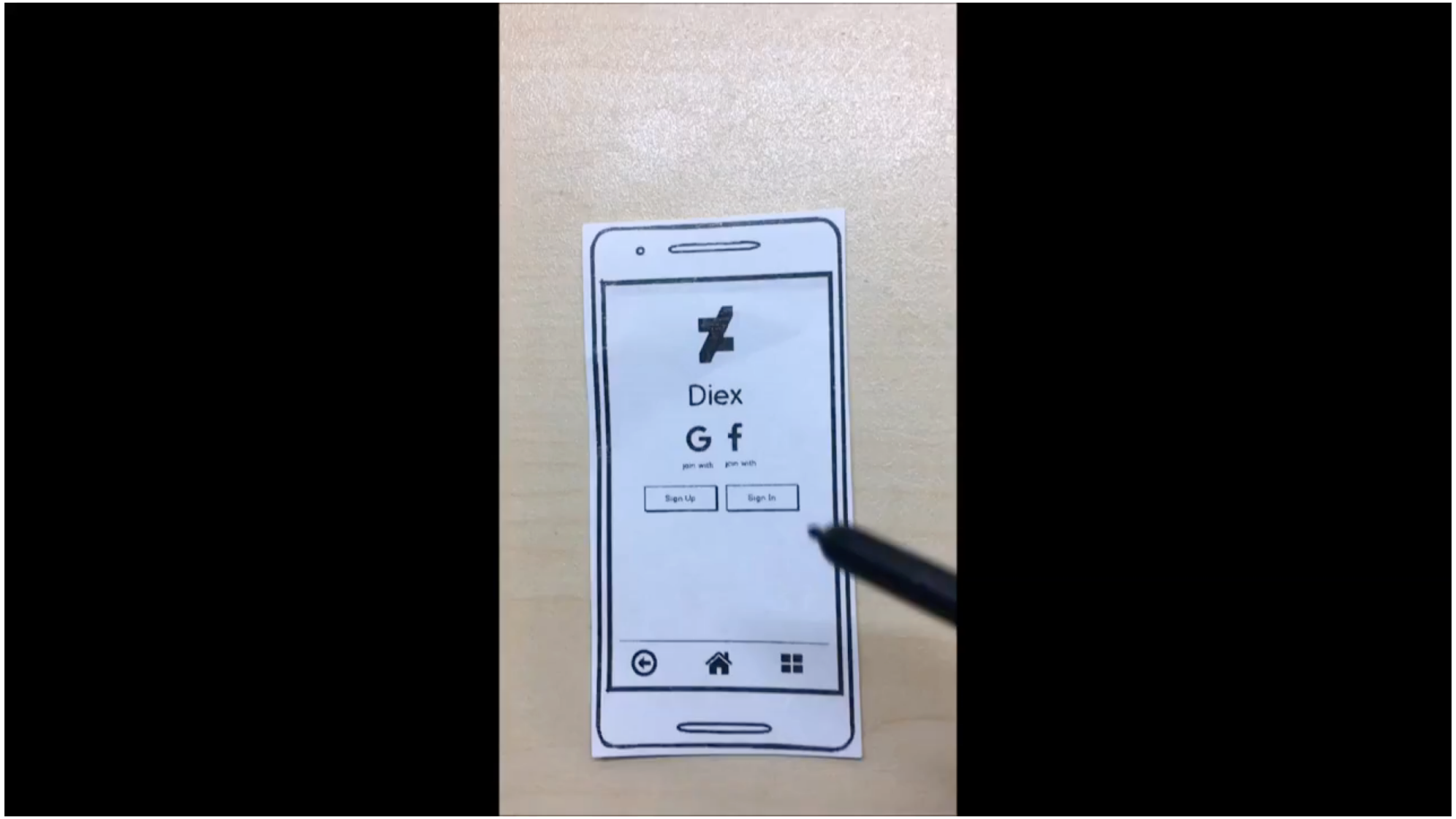
*Source: https://www.bbc.co.uk/rd/sites/50335ff370b5c262af000004/assets/540ec558acfbab59dd0e8cbf/scenario-wireframe-blog.jpg

- *Analysts and end-users' work through these scenarios to find users reactions to the system, the information they require, and how they would normally interact with the system.*

Possible Approaches to Prototyping: ‘Wizard of Oz’

- In 'Wizard of Oz' prototyping, a person simulates the responses of the system in response to some user inputs:
  - Relatively cheap as it does not require much software to be developed.
  - The user interacts with what appears to be the system but user’s inputs are actually channeled to a person who simulates the responses of the system.
  - Users get familiar with the interface, can see the interactions between interface, and the system functionality simulated by the 'Wizard of Oz’.

An Example of Wizard of Oz



Possible Approaches to Prototyping: ‘Automated Prototyping’

- A fourth generation language (4GL) is used to develop an executable prototype.
  - Example: Visual FoxPro, Visual DataFlex, PowerBuilder, SQL, Wavemaker, etc.
  - Developing an executable prototype of the system is more expensive option.
  - It involves writing software to simulate actual functionalities of the system.

Possible Approaches to Prototyping: ‘Automated Prototyping’

- Some requirements are impossible to prototype. The 'whole system' requirements are general objectives.
  - E.g., performance, efficiency, usability, reliability, etc. can vary from the final product.

- Therefore, it is wise to use prototypes only to discover the functional requirements.

Write a Draft User Manual

- Use the requirements document as a 'system specification', write an initial draft of the user manual of the system.
  - Should cover all aspects of the system functionality.
  - Make plausible assumptions about the system user interface.

- Benefits include:
  - Writing a user manual during the validation process 'forces' the detailed analysis of requirements
    - System usability issues are uncovered before system development begins.
  - Useful for users who are experimenting with a system prototype.
  - An initial draft for the actual user documentation in future.

Propose Requirements Test Cases

- For each requirement, propose one or more possible tests to check if the system meets that requirement.

- Benefits include:
  - Proposing possible tests is an effective way of revealing requirements problems – incompleteness and ambiguity.
  - The proposed set of test cases can be used as a basis for test planning and the derivation of actual test cases.

- *Do not need to be concerned with practicalities such as testing costs, avoiding redundant tests, detailed test data definition, etc.*

How to Define the Test Cases?

- To define a test case, ask these questions about a requirement:
  - What usage scenario (i.e., the context the test be applied) might be used to check the requirement?
  - Does the requirement include enough information to allow a test to be defined?
  - Is it possible to check the requirement using a single test or are multiple test cases required?
    - There may be more than one requirement embedded in a single requirement description.

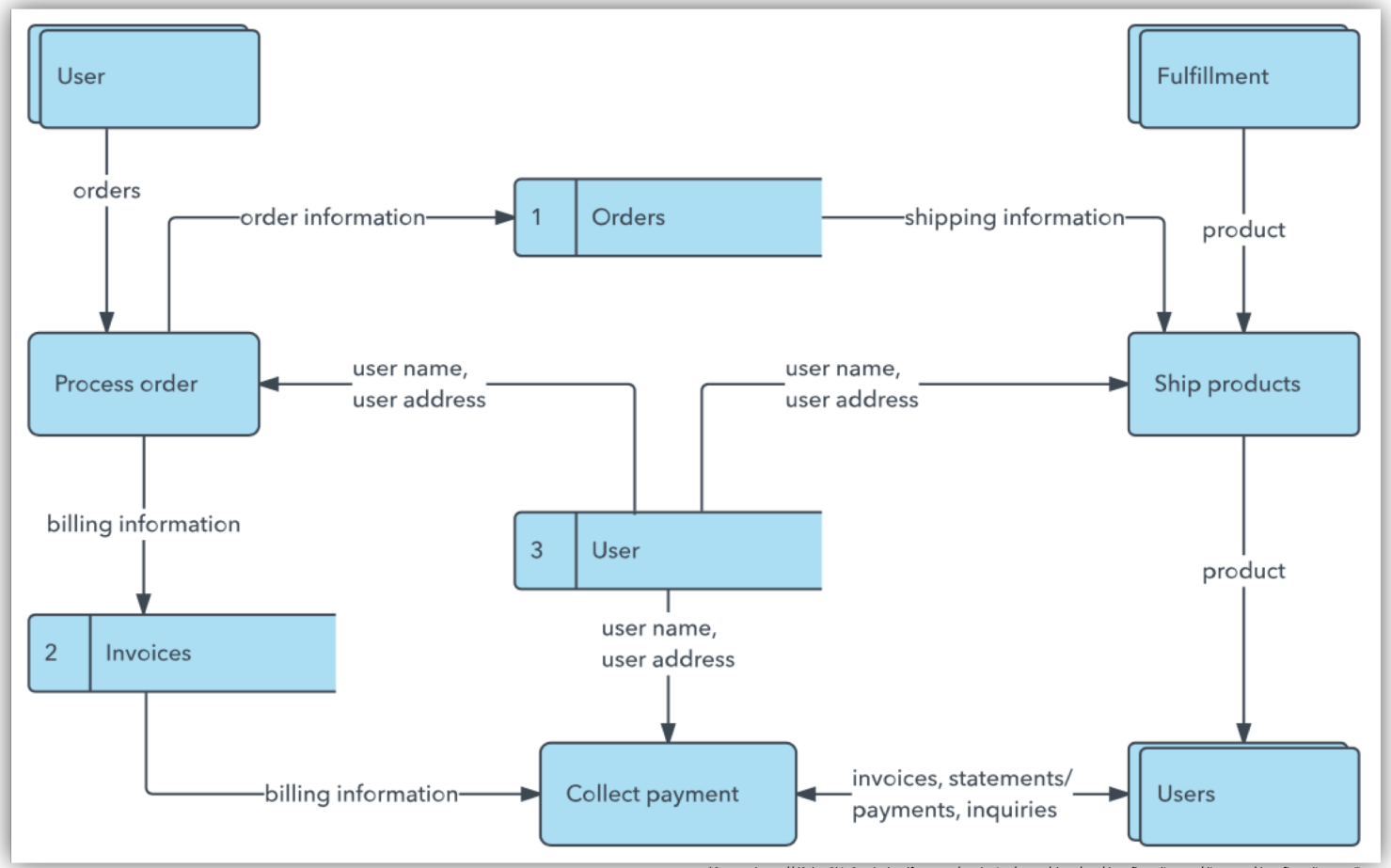
- Design a *Test Recording Form* for each requirement with the following information:
  - The requirement identifier.
  - Related requirements.
  - A brief description of the test which could be applied and why this is an objective requirements test.
  - A description of requirements problems which made test definition difficult or impossible.
  - Recommendations for addressing requirements problems which have been discovered.

Paraphrase System Models for Requirements Validation

- Convert the specification described by the graphical model or formal notation to a natural language representation.
- **Benefits include:**
  - Stakeholders, e.g., end-users, organizational management, and external stakeholders (regulators) unlikely have the time to learn the notations used to describe the system model.
  - The process of explaining the system model in natural language is an effective way to detect errors, inconsistencies, and incompleteness in the model.
- **Implementation:**
  - Use forms or tables where different components in the model are described in different fields or columns.
  - For example, in a Data Flow Diagram, use a template to describe each transformation.

Paraphrase System Models for Requirements Validation

- An example:
  - Transformation name
  - Transformation inputs and input sources the name of each input to the transformation and lists where that input comes from.
  - Transformation function what the transformation is supposed to do to convert inputs to outputs.
  - Transformation outputs the name of each output and lists where the output goes to.
  - Control any exception or control information included in the model.



*Source: https://d2slcw3kip6qmk.cloudfront.net/marketing/pages/chart/seo/data-flow-diagram/discovery/data-flow-diagram-7.svg

Requirements Management

- Requirements Validation
- Requirements Management
  - Uniquely identify each requirement
  - Define policies for requirements management
  - Define traceability policies
  - Use a database to manage requirements
  - Define change management policies
  - Identify global system requirements
  - Identify volatile requirements
  - Record rejected requirements

Requirements Management

- New requirements emerge, and existing requirements change.
  - Changing requirements must be managed to ensure the quality of the requirements.
  - System changes must be reflected as requirements changes, and vice-versa.
- Requirements management is a process which supports other RE activities and is carried out in parallel with them.

- The principal concerns of requirements management are:
  - Managing changes to agreed requirements
  - Managing the relationships between requirements
  - Managing dependencies between the requirements document and other documents produced during software engineering process.
- Maintaining requirements traceability information is the key to manage requirements:
  - A requirement is traceable if you can discover **who** suggested the requirement, **why** the requirement exists, **what** requirements are related to it and **how** that requirement relates to other information such as systems designs, implementations, and user documentation.

Uniquely Identify Requirements

- Must assign each requirement a unique identifier or reference number.
- Benefits include:
  - Make references to related requirements and to construct traceability tables.
  - Requirement identifier may serve as a primary key if stored in a database.
  - Requirements versioning in configuration management systems.
- The most commonly used approach to requirements identification is to assign numbers depending on its chapter and section in the requirements document, e.g., the 6th requirement in the 2nd section in chapter 4 would be identified as '[4.2.6]'.
- One possible problem: with similar identifiers requirements seem to be closely related.
- A good solution is to use '*mnemonic*' form derived from the contents of the requirements.

What is a Policy?

- *"A policy is a set of ideas or a plan of what to do in particular situations that has been agreed officially by a group of people, a business organization, a government, or a political party."* (Cambridge Dictionary)

- *"A policy is a set of principles, rules, and guidelines formulated or adopted by an organization to reach its long-term goals."* (www.businessdictionary.com)

Define Policies for Requirements Management (RM)

- RM policies define goals for requirements management, the procedures which should be followed, and the standards which should be conformed.

- Benefits include: Explicit policies guide people what they are expected to do and why it should be done. There will be less dependence on individual knowledge and expertise.

- **Policies *vs.* Standards?**
  - Policies set out *what* should be done; standards describe *how* the policy should be implemented in a particular situation.

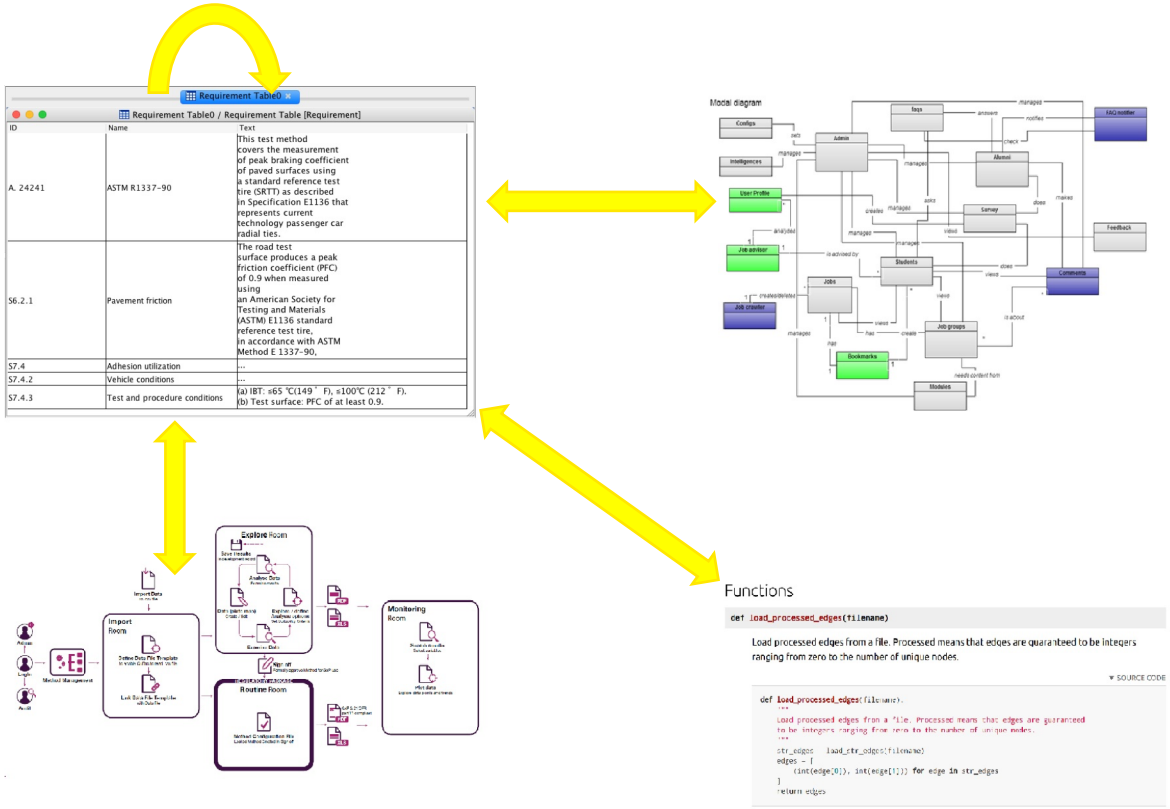
Define Policies for Requirements Management (RM)

- Requirements management policies should include:
  - A set of objectives for the RM process and rationale associated with each of these objectives.
  - The standards for requirements documents and requirements descriptions which should be used.
  - Change management and control policies for requirements.
  - Requirements review and validation policies.
  - Traceability policies defining what information on dependencies between requirements should be maintained and how this information should be used and managed.
  - The criteria when these policies can be ignored
    - Managers use their own judgement on how to implement a requirements change.

Define Requirements Traceability Policies

- Define what traceability information should be maintained, how this should be represented.

- Traceability information allows you to find dependencies:
  - between requirements
  - between the requirements and the system design
  - between the requirements and the components
  - between the requirements and the documentation



Traceability Information

- There are different types of traceability information to maintain:

Traceability Types	Description
Requirements-Sources	Links the requirement and the people (or documents) who (or that) specified the requirement.
Requirements-Rationale	Links the requirement with a description of why that requirement has been specified.
Requirements-Requirements	Links requirements with other requirements which are, in some way, dependent on them.
Requirements-Architecture	Links requirements with the sub-systems where these requirements are implemented. Crucial when sub-systems are being developed by different sub-contractors.
Requirements-Design	Links requirements with specific components in the system which are used to implement the requirement. These may be hardware or software components.
Requirements-Interface	Links requirements with the interfaces of external systems which are used in the provision of the requirements.

Basic Techniques to Maintain Traceability Information

- **Traceability Table:** In a cross-reference matrix, traceability table shows the relationships between requirements or between requirements and design components.

- Read the table column-wise.

- Extend the traceability table with the types of relationships.
  - specifies / is_specified_by
  - requires / is_required_by
  - constraints / is_constrained_by

- Suitable for relatively small number of requirements.

	Depends-on			
	R1	R2	R3	R4
R1			X	
R2				
R3		X		
R4	X		X	

	Depends-on			
	R1	R2	R3	R4
R1			specifies	
R2				
R3		constraints		
R4	requires		constraints	

Basic Techniques to Maintain Traceability Information

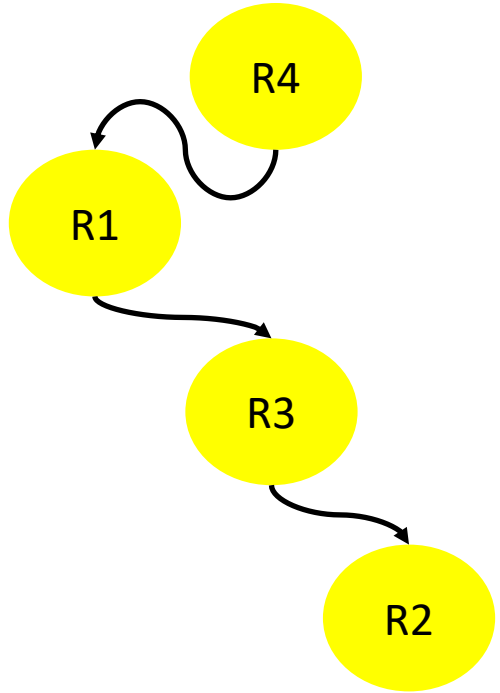
- **Traceability List** (manual or automated): A simplified form of traceability table where, for each requirement, one or more lists of the identifiers of related requirements are placed.

- Traceability lists are:
  - More compact than traceability tables and manageable with large numbers of requirements
  - Less prone to error than traceability tables.

- Use several lists, one for each type of relationship.

- **Disadvantage:** no easy way to assess the inverse of a relationship.

Requirement	Depends-on
R1	R4
R2	R3
R3	R1, R4
R4	-



Define a Database to Manage Requirements

- Maintain a requirements database and store the individual requirements.
- **Benefits include:**
  - Easy to maintain links between individual requirements, to search, and abstract related groups of requirements.
  - The requirements may be automatically processed to extract particular types of information, i.e., generate traceability tables and lists automatically from.

- Consider these things before opting for a database:

§ How are requirements **expressed**?

§ How **many** requirements do you typically need to manage?

§ Are the requirements always developed/managed by teams working at the **same site** using the same types of computers?

§ Do you **already** use a database for software engineering support?

§ What in-house database **expertise** do you have available?

- *The relational databases are most commonly used type of database. However, object-oriented databases are developed recently, structurally more suited to requirements management.*

Define Change Management Policy

- Define policies for managing changes to requirements – how changes are formally proposed, analyzed, and reviewed.
- Similar information is collected for each proposed change, then judgements are made about the costs and benefits of proposed changes.
- Benefits include:
  - Keep track of the status of these changes and to produce reports on requirements evolution.
  - Give stakeholders a formal mechanism for proposing changes to requirements – not favoring specific stakeholders.
- The change management policies should define:
  - The change request process and the information required to process each change request.
  - The process used to analyze the impact and costs of change and the associated traceability information.
  - The membership of the body that formally considers change requests.
  - The software support for the change control process -- a change management database.

Identify Global System Requirements (GSRs)

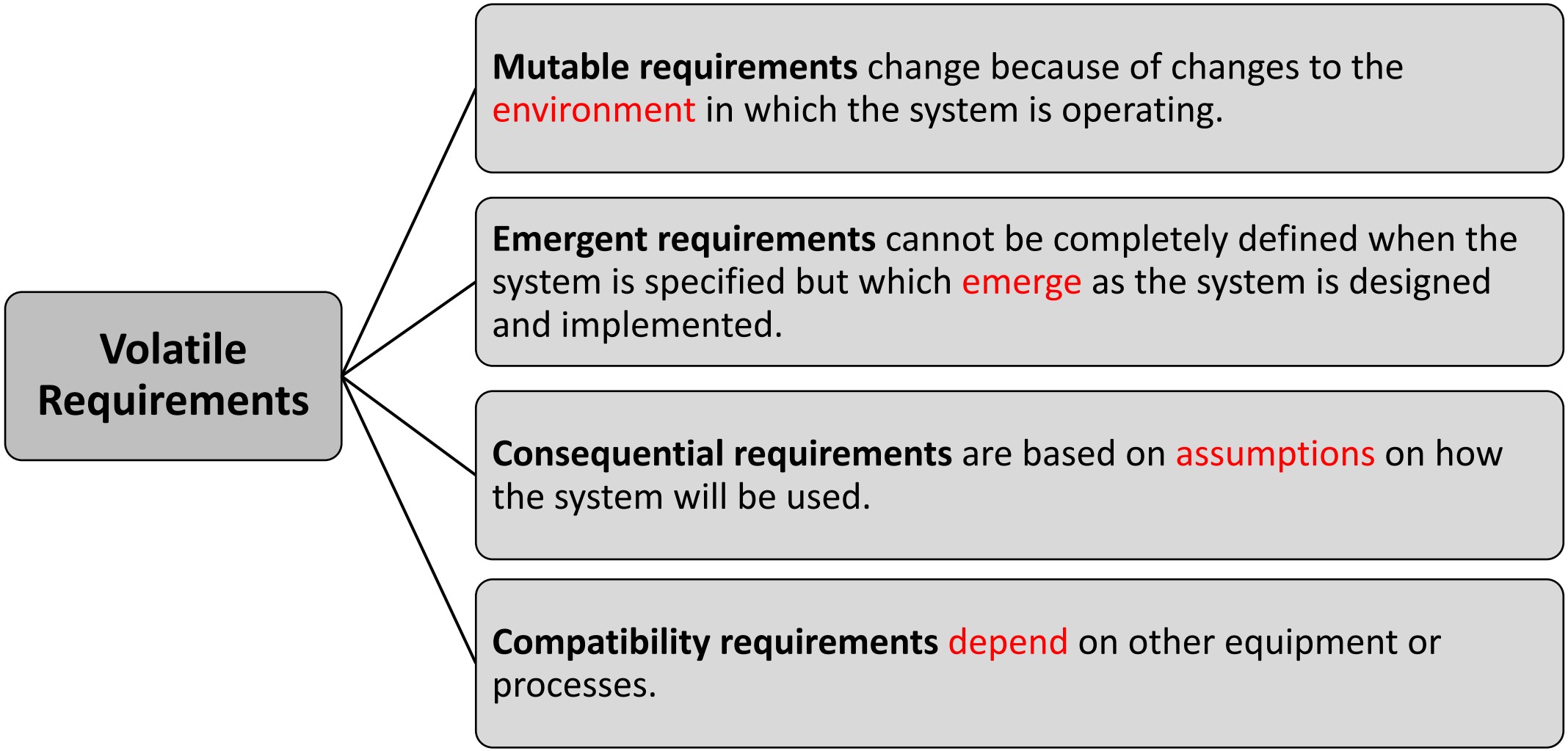
- Requirements that are desirable or essential properties of the 'system as a whole'.
- Identify these requirements during the requirements analysis and the validation process.
- Benefit: Global requirements are expensive to change. If identified and analysed in advance, it will minimize future requirements changes due to omissions and misunderstandings.
- Ask the following questions for each requirement to identify global requirements:
  1. Is the requirement expressed in a very **general** way? **YES**
  2. Does the requirement express some global, **non-functional** system characteristic? **YES**
  3. Is the requirement a general **user interface** requirement? **YES**
  4. Does the requirement refer to **specific system functionality** or a specific system service that must be provided? **NO**
  5. Can the requirement be **mapped** onto part of the system model? **NO**
  6. Does the requirement refer to **specific data** or components of the system? **NO**

- An iterative process, i.e., will not have the identification of global requirements right first time.

Identify Volatile Requirements

- Maintain a list of the most volatile requirements that are most likely to change.
  - Predict likely changes to these requirements.
  - Experienced engineers should examine all of the system requirements and identify volatile requirements.
- Benefits include:
  - Developers design the system so that volatile requirements are implemented in modules which are loosely coupled to the rest of the system, i.e., software changes to accommodate requirements change will be easier to implement.
  - Planning for change is easier if likely changes are predicted.
- Update the list of volatile requirements over time; as some of the requirements which were set volatile could be fairly stable.
- In a requirements database, add a field to the requirements record whose value indicates the estimated volatility of the requirement. Use a three-point scale: *1 = 'fairly stable'*, *2 = 'subject to medium term change'*, and *3 = 'requirement is likely to change in the short-term'*.

Types of Volatile Requirements



Record Rejected Requirements

- Keep a record of requirements which have been proposed and subsequently rejected after analysis or negotiation, and why.
- Benefits include:
  - It is fairly common for rejected requirements to be proposed again because there was, presumably, a good reason for proposing them.
  - It is easy to forget why requirements got rejected. If maintain a record, we can avoid some or all of the costs of repeating the requirements analysis.
- In the requirements database, include a status field where a requirement is marked as '*accepted*', '*under consideration*', '*rejected*', and a link to the reasons why the requirement was rejected.

- *As part of the process of assessing new requirements, always consult the database of rejected requirements to see if similar requirements were rejected.*

Key Takeaways

1. The objectives of requirements validation are to check the set of identified requirements and discover possible problems with the requirements.

2. System requirements should be validated by a group of people who systematically check the requirements, meet to discuss problems with the requirements, and agree on how these problems should be fixed.

3. Requirements document should be reviewed by a multi-disciplinary team drawn from people with different backgrounds.

4. Implement a prototype of the system for experimentation. Stakeholders can try the system to see if it meets their real needs and can suggest improvements.

6. Each requirement should be assigned a **unique identifier** or reference number. One good solution is to use '**mnemonic**' form derived from the contents of the requirements.

7. Requirements management policies define goals for requirements management, the procedures which should be followed, and the standards which should be used.

8. Define policies for managing changes to requirements – how changes are formally proposed, analyzed, and reviewed.

9. Maintain a list of the most volatile requirements that are most likely to change. Keep a record of requirements which have been proposed and subsequently rejected after analysis or negotiation, and why.

Linnéuniversitetet

Questions?