Software Engineering

Lecture 03

Requirement Engineering

Requirements Engineering

- □ Requirement: A function, constraint or other property that the system must provide to fill the needs of the system’s intended user(s)
- □ Engineering: implies that systematic and repeatable techniques should be used
- □ Requirement Engineering means that requirements for a product are defined, managed and tested systematically

Requirements Engineering

- It is essential that the software engineering team understand the requirements of a problem before the team tries to solve the problem.
- In some cases requirements engineering may be abbreviated, but it is never abandoned.
- RE is software engineering actions that start with communication activity and continues into the modeling activity.
- RE establishes a solid base for design and construction. Without it, resulting software has a high probability of not meeting customer needs.

Characteristics of a Good Requirement

- Clear and Unambiguous
  - standard structure
  - has only one possible interpretation
  - Not more than one requirement in one sentence
- Correct
  - A requirement contributes to a real need
- Understandable
  - A reader can easily understand the meaning of the requirement
- Verifiable
  - A requirement can be tested
- Complete
- Consistent
- Traceable

Why is Getting Good Requirements Hard?

- Stakeholders don’t know what they really want.
- Stakeholders express requirements in their own terms.
- Different stakeholders may have conflicting requirements.
- Organisational and political factors may influence the system requirements.
- The requirements change during the RE process. New stakeholders may emerge and the business environment change.

Types of Requirements

- **Functional requirements**
  - Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
  - May state what the system should not do.
- **Non-functional requirements**
  - Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
  - Often apply to the system as a whole rather than individual features or services.
- **Domain requirements**
  - Constraints on the system from the domain of operation

Functional Requirements

- Describe functionality or system services.
- Depend on the type of software, expected users and the type of system where the software is used.
- Functional user requirements may be high-level statements of what the system should do.
- Functional system requirements should describe the system services in detail.
- Essentially, these are the ‘whats’ of the system that we often refer to. These are not ‘all that there is,’ but these should describe the overall functionality of the system.

Non-functional Requirements

- These define system properties and constraints e.g. reliability, response time, maintainability, scalability, portability, and storage requirements.
- Constraints are I/O device capability, system representations, etc.
- Process requirements may also be specified mandating a particular IDE, programming language or development method.
- (Often internal to an organization or required for fit / compatibility with other comparable systems.)
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system may be useless.

Non-functional Requirements Implementation

- Non-functional requirements may affect the overall architecture of a system rather than the individual components.
  - For example, to ensure that performance requirements are met, you may have to organize the system to minimize communications between components.
- A single non-functional requirement, such as a security requirement, may generate a number of related functional requirements that define system services that are required.
  - It may also generate requirements that restrict existing requirements.

Metrics for specifying nonfunctional requirements

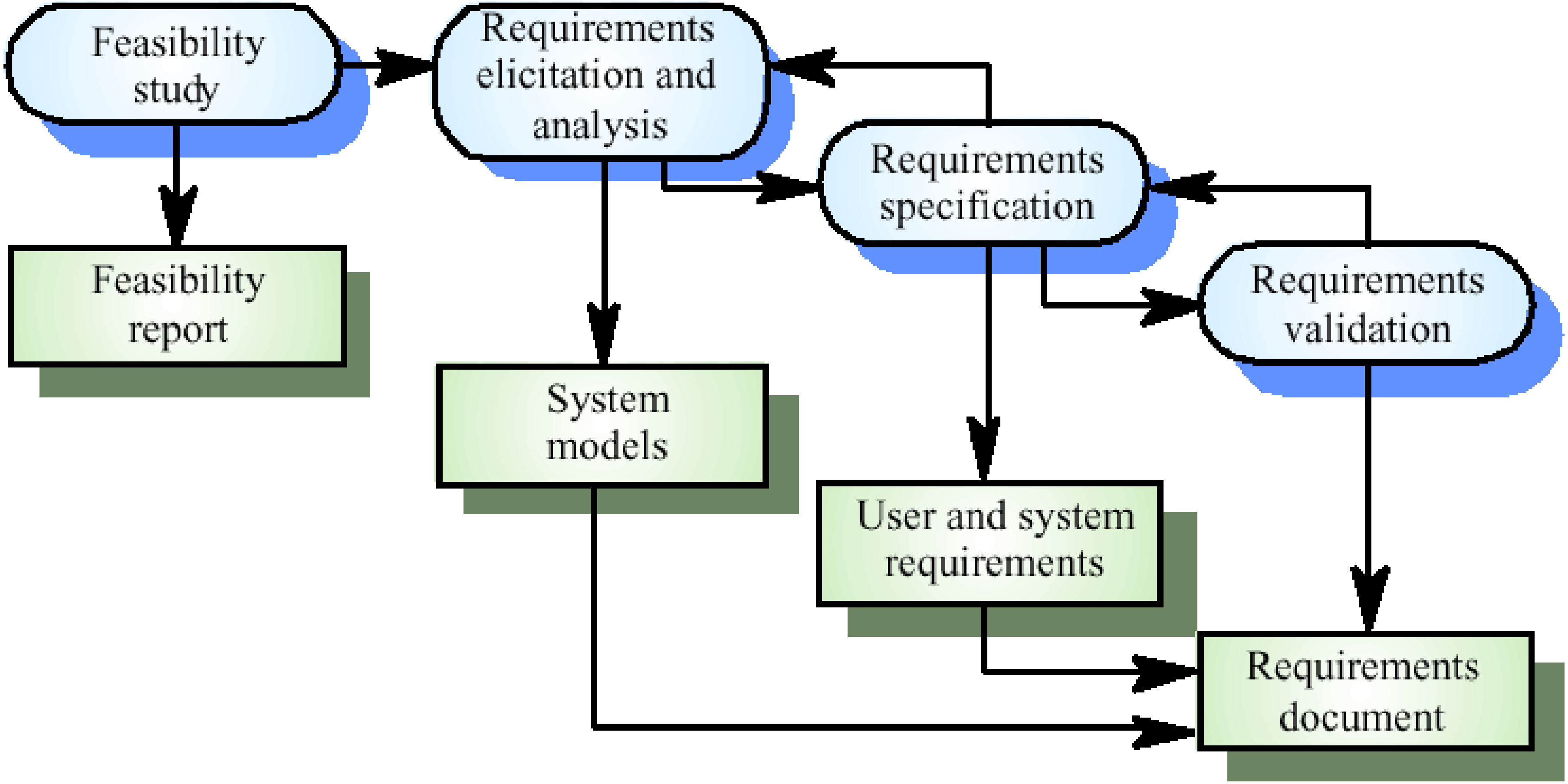
Property	**Measure**
Speed	Processed transactions/second User/event response time Screen refresh time
Size	Mbytes Number of ROM chips
Ease of use	Training time Number of help frames
Reliability	**Mean time to failure (MTTF)** Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure **(MTTR)** Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

- **User requirements**
  - Statements in natural language plus diagrams of the services the system provides and its operational constraints.
  - Written for customers.
- **System requirements**
  - A structured document setting out detailed descriptions of the system’s functions, services and operational constraints.
  - Defines what should be implemented so may be part of a contract between client and contractor.
  - Whom do you think these are written for?
  - These are higher level than functional and non-functional requirements, which these may subsume.

Requirements engineering processes

- Requirements elicitation
- Requirements analysis
- Requirements validation
- Requirements management

The requirements engineering process



Requirements Engineering Tasks

- Inception —Establish a basic understanding of the problem and the nature of the solution.
- Elicitation —Draw out the requirements from stakeholders.
- Elaboration (Highly structured)—Create an analysis model that represents information, functional, and behavioral aspects of the requirements.
- Negotiation—Agree on a deliverable system that is realistic for developers and customers.
- Specification—Describe the requirements formally or informally.
- Validation —Review the requirement specification for errors, ambiguities, omissions, and conflicts.
- Requirements management —Manage changing requirements.

Inception

- Inception— Ask “context-free” questions that establish …
  - Basic understanding of the problem
  - The people who want a solution
  - The nature of the solution that is desired, and
  - The effectiveness of preliminary communication and collaboration between the customer and the developer

Elicitation

- Elicitation - elicit requirements from customers, users and others.
  - Find out from customers, users and others what the product objectives are
  - what is to be done
  - how the product fits into business needs, and
  - how the product is used on a day to day basis

Why Requirement elicitation is difficult?

- Problems of scope:
  - The boundary of the system is ill-defined.
  - Customers/users specify unnecessary technical detail that may confuse rather than clarify objectives.
- Problem of understanding:
  - Customers are not completely sure of what is needed.
  - Customers have a poor understanding of the capabilities and limitations of the computing environment.
  - Customers don’t have a full understanding of their problem domain.
  - Customers have trouble communicating needs to the system engineer.
  - Customers omit detail that is believed to be obvious.
  - Customers specify requirements that conflict with other requirements.
  - Customers specify requirements that are ambiguous or not able to test.
- Problems of volatility:
  - Requirement change over time.

Initiating Requirements Engineering Process

- □ **Identify stakeholders**
  - Stakeholder can be "anyone who benefits in a direct or indirect way from the system which is being developed"

  Ex. Business manager, project manager, marketing people, software engineer, support engineer, end-users, internal-external customers, consultants, maintenance engineer.
  - Each one of them has different view of the system.
- □ **Recognize multiple points of view**
  - Marketing group concern about feature and function to excite potential market. To sell easily in the market.
  - Business manager concern about feature built within budget and will be ready to meet market.
  - End user – Easy to learn and use.
  - SE – product functioning at various infrastructure support.
  - Support engineer – Maintainability of software.

Role of RE is to categorize all stakeholder information in a way that there could be no inconsistent or conflict requirement with one another

Initiating Requirements Engineering Process (cont.)

- **Work toward collaboration**
  - RE identify areas of commonality (i.e. Agreed requirement) and areas of conflict or inconsistency.
  - It does not mean requirement defined by committee. It may happened they providing just view of their requirement.
  - Business manager or senior technologist may make final decision.
- **Asking the first questions**
  - Who is behind the request for this work?
  - Who will use the solution?
  - What will be the economic benefit of a successful solution
  - Is there another source for the solution that you need?

These questions will help – stakeholder interest in the software & measurable benefit of successful implementation.

Asking the question

Next set of questions – better understanding of the problem.

- What business problem (s) will this solution address?
- Describe business environment in which the solution will be used?
- will performance or productivity issues affect the solution is approached?

Final set of questions – Effectiveness of communication

- Are my questions relevant to the problem?
- Am I asking too many questions?
- Can anyone else provide additional information?
- should I be asking you anything else?

Eliciting Requirement

Approach for eliciting requirement:

- □ Collaborative Requirement Gathering
- □ Quality Function Deployment
- □ User Scenarios
- □ Elicitation Work Products

Collaborative Requirement Gathering

- Meetings are attended by all interested stakeholders.
- Rules established for preparation and participation.
- Agenda should be formal enough to cover all important points, but informal enough to encourage the free flow of ideas.
- A facilitator controls the meeting.
- A definition mechanism (blackboard, flip charts, etc.) is used.
- During the meeting:
  - The problem is identified.
  - Elements of the solution are proposed.
  - Different approaches are negotiated.
  - A preliminary set of solution requirements are obtained.
  - The atmosphere is collaborative and non-threatening.
- Flow of event – Outline the sequence of events occurs
  - Requirement gathering meeting (initial meeting)
  - During meeting
  - Follow the meeting.

Collaborative requirement gathering (contd.)

- In initial meeting, distribute "Product request" (defined by stakeholder) to all attendee.
- Based on product request, each attendee is asked to make
  - List of objects (Internal or external system objects)
  - List of services(Processes or functions)
  - List of constraints (cost, size, business rules) and performance criteria(speed, accuracy) are developed.
- Collect lists from everyone and combined.
- Combined list eliminates redundant entries, add new ideas , but does not delete anything.
- Objective is to develop a consensus list in each topic area (objects, services, constraints and performance).
- Based on lists, team is divided into smaller sub-teams : each works to develop mini-specification for one or more entries on each of the lists.

Collaborative requirement gathering (Contd.)

- Each sub-team the presents its mini-specification to all attendees for discussion. Addition, deletion and further elaboration are made.
- Now each team makes a list of validation criteria for the product and present to team.
- Finally, one or more participants is assigned the task of writing a complete draft specification.

Quality Function Deployment

- It is a technique that translate the needs of the customer into technical requirement for software.
- Concentrates on maximizing customer satisfaction.
- QFD emphasizes – what is valuable to the customer and then deploys these values throughout the engineering process.

Three types of requirement:

1. Normal Requirements – reflect objectives and goals stated for product. If requirement are present in final products, customer is satisfied.
2. Expected Requirements – customer does not explicitly state them. Customer assumes it is implicitly available with the system.
3. Exciting Requirements- Features that go beyond the customer's expectation.

During meeting with customer –

Function deployment determines the "value" of each function required of the system.

Information deployment identifies data objects and events and also tied with functions.

Task deployment examines the behavior of the system.

Value analysis determines the priority of requirements during these 3 deployments.

User Scenario

- It is difficult to move into more software engineering activities until s/w team understands how these functions and features will be used by diff. end-users.
- Developers and users create a set of usage threads for the system to be constructed
- A use-case scenario is a story about how someone or something external to the software (known as an actor) interacts with the system.
- Describe how the system will be used
- Each scenario is described from the point-of-view of an “actor”—a person or device that interacts with the software in some way

Elicitation Work Products

Elicitation work product will vary depending upon the size of the system or product to be built.

- Statement of need and feasibility.
- Statement of scope.
- List of participants in requirements elicitation.
- Description of the system’s technical environment.
- List of requirements and associated domain constraints.
- List of usage scenarios.
- Any prototypes developed to refine requirements.

Elaboration

- Focuses on developing a refined technical model of software functions, features, and constraints using the information obtained during inception and elicitation
- Create an analysis model that identifies data, function and behavioral requirements.
- It is driven by the creation and refinement of user scenarios that describe how the end-user will interact with the system.
- Each event parsed into extracted.
- End result defines informational, functional and behavioral domain of the problem

Negotiation

- Negotiation - agree on a deliverable system that is realistic for developers and customers
  - Requirements are categorized and organized into subsets
  - Relations among requirements identified
  - Requirements reviewed for correctness
  - Requirements prioritized based on customer needs
  - Negotiation about requirements, project cost and project timeline.
  - There should be no winner and no loser in effective negotiation.

Specification

- Specification – Different things to different people.
- It can be –
  - Written Document
  - A set of graphical models,
  - A formal mathematical models
  - Collection of usage scenario.
  - A prototype
  - Combination of above.
- The Formality and format of a specification varies with the size and the complexity of the software to be built.
- For large systems, written document, language descriptions, and graphical models may be the best approach.
- For small systems or products, usage scenarios

Specification Principles

- May be viewed as representation process.

1. Separate functionality from implementation.
2. Develop a model of the desired behavior of a system.
3. Establish the context in which software operates by specifying the manner.
4. Define the environment in which the system operates and indicate how.

Specification Principles (cont.)

5. Create a cognitive model rather than a design or implementation model. The cognitive model describes a system as perceived by its user community.
6. The specifications must be tolerant of incompleteness and augmentable.
7. Establish the content and structure of a specification in a way that will enable it to be amenable to change.

Specification Representation

- □**Representation format and content should be relevant to the problem.**
  - For example, a specification for a manufacturing automation system might use different symbology, diagrams and language than the specification for a programming language compiler.
- □**Information contained within the specification should be nested (layered).**
  - Paragraph and diagram numbering schemes should indicate the level of detail that is being presented.
  - It is sometimes worthwhile to present the same information at different levels of abstraction to aid in understanding.
- □**Diagrams and other notational forms should be restricted in number and consistent in use**.
  - Confusing or inconsistent notation, whether graphical or symbolic, degrades understanding and fosters errors.
- □**Representations should be revisable.**

Software Requirements Specification

- It contains a complete information description, a detailed functional description, a representation of system behavior, an indication of performance requirements and design constraints, appropriate validation criteria, and other information pertinent to requirements.

Format of SRS:

Introduction of the software requirements specification states the goals and objectives of the software, describing it in the context of the computer-based system.

Information content, flow, and structure are documented. Hardware, software, and human interfaces are described for external system elements and internal software functions.

Functional Description A processing narrative is provided for each function, design constraints are stated and justified & performance characteristics are stated

Behavioral Description operation of the software as a consequence of external events and internally generated control characteristics.

Software Requirements Specification (Cont.)

Validation Criteria is probably the most important and, ironically, the most often neglected section of the *Software Requirements Specification (SRS). Testing or validating each user-scenario.*

Finally, the specification includes a ***Bibliography and Appendix****.* The *bibliography* contains references to all documents that relate to the software. The *appendix* contains information that supplements the specifications

Validation

- Requirements Validation - formal technical review mechanism that looks for
  - Errors in content or interpretation
  - Areas where clarification may be required
  - Missing information
  - Inconsistencies (a major problem when large products or systems are engineered)
  - Conflicting or unrealistic (unachievable) requirements.

Requirements validation techniques

- Requirements reviews
  - Systematic manual analysis of the requirements
- Prototyping
  - Using an executable model of the system to check requirements.
- Test-case generation
  - Developing tests for requirements to check testability
- Automated consistency analysis
  - Checking the consistency of a structured requirements description

Requirements Review

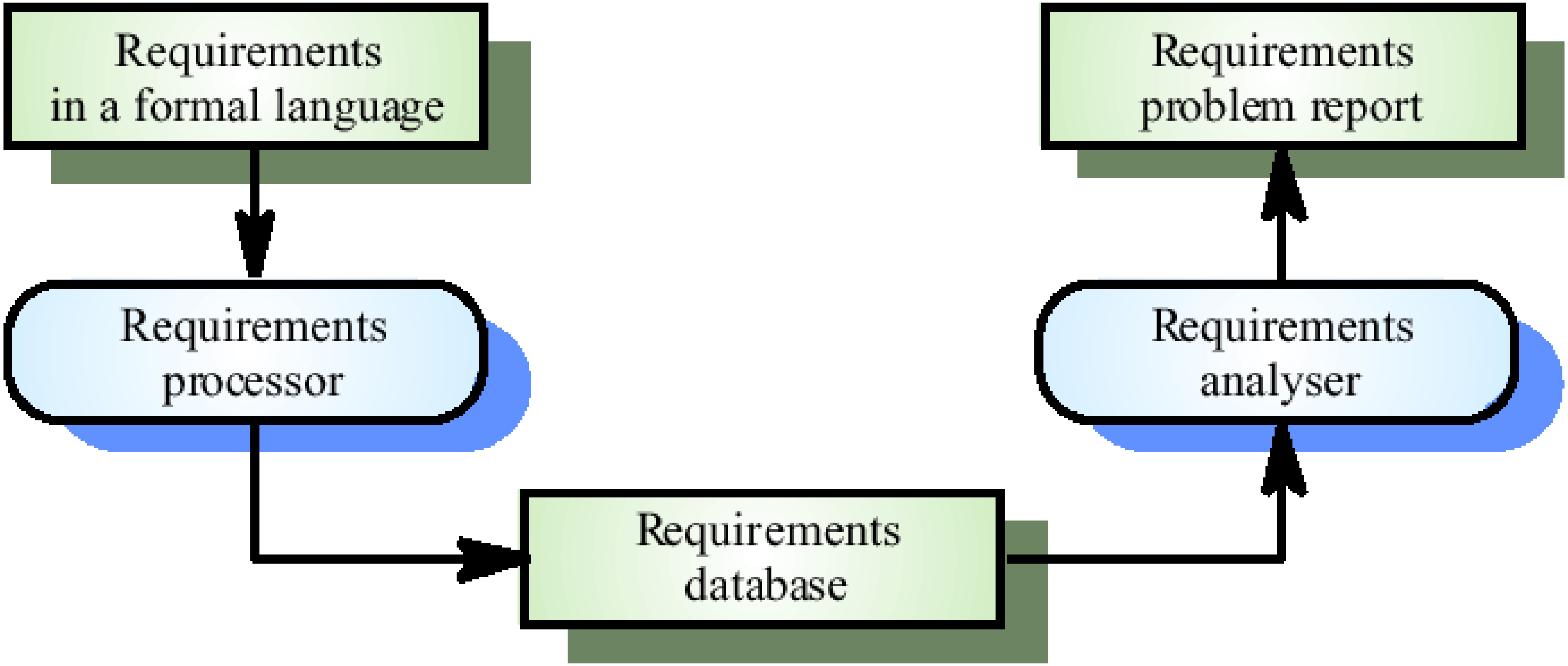
- A review of the *SRS* (and/or prototype) is conducted by both the software developer and the customer.
- Conducted at a macroscopic level
  - Ensure that specification is complete
  - Consistent
  - Accurate (Information, functional and behavioral domain considered).
- Review becomes more detailed while examining Information, functional and behavioral domain.
- Examining not only broad descriptions but the way in which requirement worded.

E.g. Terms like “Vague ” (some, sometimes, often, usually) should be flag by reviewer for further clarification.

Requirements Review (cont.)

- Once review is complete – SRS “signed off” by both customer and developer. (“contract” for software development)
- Requests for changes in requirements after the specification is finalized will not be eliminated.
- Change is an extension of software scope and therefore can increase cost and/or delivery of product.
- During the review, changes to the specification may be recommended.
- It can be extremely difficult to assess the global impact of a change; that is, how a change in one function affects requirements for other functions

Automated consistency checking



Requirements in a formal language

Requirements processor

Requirements database

Requirements analyser

Requirements problem report

Requirement Management

- Set of activities that help project team to identify, control, and track requirements and changes as project proceeds
- Requirements begin with identification. Each requirement is assigned a unique identifier. Once requirement have been identified, traceability table are developed.

Traceability Table:

- **Features traceability table** - shows how requirements relate to customer observable features
- **Source traceability table** - identifies source of each requirement
- **Dependency traceability table** - indicate relations among requirements
- **Subsystem traceability table** - requirements categorized by subsystem
- **Interface traceability table** - shows requirement relations to internal and external interfaces

It will help to track, if change in one requirement will affect different aspects of the system.