Group 8

Syed Khandker, Esko Reivari

Sandesh Hyoju, Timo Tammi & Henri Murtonen

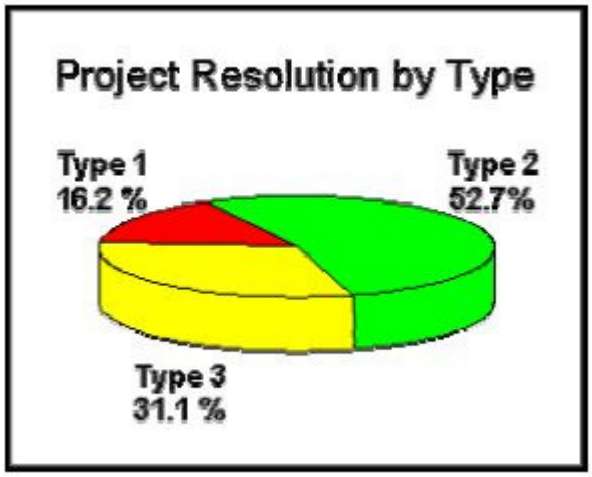
1st & 2nd phase of the ITKS452 Diary (part 1& part2)

1. Using CHAOS report's findings (or similar), explain why requirements are important. Also explain how requirements help to increase the chances of having a successful project.

The Standish Group conducted four focus groups and numerous personal interviews to provide qualitative context for the survey results of software failures. For purposes of the study, projects were classified into three resolution types:

- Resolution Type 1, or project success: The project is completed on-time and on-budget, with all features and functions as initially specified.
- Resolution Type 2, or project challenged: The project is completed and operational but over-budget, over the time estimate, and offers fewer features and functions than originally specified.
- Resolution Type 3, or project impaired: The project is cancelled at some point during the development cycle.

Overall, the success rate was only 16.2%, while challenged projects accounted for 52.7%, and impaired (cancelled) for 31.1%.



That's why requirements are important to finish project on time, on budget and with all features and function that are initially specified.

Requirements help to increase the chances of having a successful project as it is the process of establishing the services that the customer requires from a system and the constraints under which it operates and is developed. The major reasons that a project will succeed are user involvement, proper planning, executive management support and a clear vision and clear statement of requirements.

2. What beliefs are often hindering good requirements practices? Why they are actually misbelieves?

The beliefs which are often hindering good requirements practices are:

- It is not worth discovering requirements directly from users: e.g:
    - We have developed these kind of products for a long time and we know user needs.
    - We developers also use our own product and therefore we can act as users.
    - We are developing a new product and therefore users cannot have any requirements for it.

- It is difficult to discover requirements directly from users: e.g.:
    - Users are unable to say what they need and want.
    - There are so many users that we cannot interview them all.

- It is risky to discover requirements directly from users: e.g:
    - We cannot show our customers that we do not know the basics of their business.
    - Product development personnel can spoil customer relationships by asking stupid questions.

- It is not worth documenting user requirements systematically: e.g:
    - Our customers are interested in documents presenting technical features of software, not in documents presenting what they know anyway.
    - We do not have time for user requirements documentation.

They are actually misbelieves because:

- The more deeply user needs are understood, the more useful and usable systems can be developed.
- Users are experts in their tasks, and therefore they are the primary source of real user requirements (domain knowledge).
- For a new product, users hardly can say what they want; but they will quickly recognize something that they do not want.
- There are elicitation (esille saaminen) techniques like interviewing and observing that product development engineers can learn and use easily.
- By combining interview and observation techniques it is possible to get a comprehensive picture of user needs without requiring users to articulate their needs explicitly.
- Better to interview a small group of users than not to do this at all
- Well-prepared user visits can improve a company's image among its customers and create competitive edge.

- When this lack of knowledge is discovered at a later stage, consequences will be worse
- Requirements document the agreement about the system, not just the content of customers' heads.
- Well-documented user requirements are useful information for many groups: designers, testers, user manual writers, management and marketing personnel.
- Systematic user requirement documentation at the beginning of the product development saves time and decreases rework in later phases of the project.

Misconceptions about requirements engineering can strongly influence a company's processes. Many companies and organizations have a solid understanding of requirements processes, but some do not. Some of the more common misconceptions are listed under the headings that follow.

- Misconception 1: Any Subject Matter Expert Can Become a Requirements Engineer after a Week or Two of Training Requirements engineers need strong communication and knowledge of engineering skills, the ability to organize and manage a data set of requirements, high-quality written and visual presentation skills, and the ability to extract and model business processes using both text and graphical (e.g., Integration DEFinition [IDEF], Unified Modeling Language [UML]) techniques. First and foremost, to elicit requirements from stakeholders requires the ability to interact with a variety of roles and skill levels, from subject matter experts (detailed product requirements) to corporate officers (elicitation of business goals). Moreover, people have to be trained to write good specifications.
- *Misconception 2: Nonfunctional and Functional Requirements Can Be Elicited Using Separate Teams and Processes* The subject domains for nonfunctional and functional requirements are related, may impact each other, and may result in iterative changes as work progresses. Team isolation may do more harm than good.
- Misconception 3: Processes That Work for a Small Number of Requirements Will Scale Requirements engineering processes do not scale well unless crafted carefully. For example, a trace matrix is an N × N matrix, where N is the number of requirements of interest. In each cell, a mark or arrow indicates that there is a trace

from requirement Ri(row i) to requirement Rj (column j). It is relatively easy to inspect, say, a 50-requirement matrix, but what happens when five to ten thousand requirements are needed to define a product? Filtering and prioritization become important in order to retrieve results that can be better understood, but the requirement annotations necessary to provide such filtering are often neglected up front because the database is initially small

3. Provide some definitions for and explain such concepts as "requirements", "requirements engineering", "requirements development", "requirements management", "stakeholders".

Requirement

A requirement is a singular documented physical and functional need that a particular design, product or process must be able to perform. It is most commonly used in a formal sense in system engineering, software engineering or enterprise engineering. It is a statement that identifies a necessary attribute, capability, characteristic or quality of a system for it to have value and utility to a customer, organization, internal user, or other stakeholder.

It may range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification. This is inevitable as requirements may serve a dual function

- May be the basis for a bid for a contract -therefore must be open to interpretation
- May be the basis for the contract itself - therefore must be defined in detail
- Both these statements may be called requirements

Requirement Engineering

Requirements engineering is a systematic way of developing requirements through an iterative process of analyzing a problem, documenting the resulting observations, and checking the accuracy of the understanding gained.

- Requirements engineering is comprised of two major tasks: analysis and modeling
- The requirements themselves are the descriptions of the system services and constraints that are generated during the requirements engineering process.

Requirement Development

Requirement development (RD) is the part of the project during which the needs of the customer are gathered and translated into a specification of what the system must do. RD can further subdivide requirements development into elicitation, analysis, specification and validation. These sub disciplines encompass all the activities involved with gathering, evaluating and documenting the requirements for a software including the following:

- Identifying the product's expected user classes
- Eliciting needs from individuals who represent each user class
- Understanding user tasks and goals and the business objective with which those tasks align.
- Analyzing the information received from users to distinguish their task goals from functional requirements, non functional requirements, business rules, suggested solutions and extraneous information.
- Understanding the relative importance of quality attributes
- Translating the collected user needs into written requirements specifications and models.

Requirement Management

Requirements management is the process of documenting, analyzing, tracing, prioritizing and agreeing on requirements and then controlling change and communicating to relevant stakeholders. It is a continuous process throughout a project.

Stakeholder

A stakeholder is a person or organization who influences a system's requirements or who is impacted by that system.

Stakeholders can affect or be affected by the organization's actions, objectives and policies. Some examples of key stakeholders :

- Customers who fund a project or acquire a product to satisfy their organization's business objectives
- Users who interact directly or indirectly with the product
- Requirements analysts who write the requirements and communicate them to the development community
- Developers who design, implement and maintain the produc
- Tester who determine whether the product behaves as intended.
- Documentation writers who produce user manuals, training materials and help systems.
- Project managers who plan the project and guide the development team to a successful delivery
- Sales, marketing, field support, help desk and other people who will have to work with the product and its customer

4. Describe the machine-environment model (so called Jackson model).

Jackson system development model is a linear software development methodology developed by Michael A. Jackson.

Three basic principles of operation of Jackson model is that:

- Development must start with describing and modelling the real world, rather than specifying or structuring the function performed by the system. A system made using Jackson model method performs the simulation of the real world before any direct attention is paid to function or purpose of the system.

- An adequate model of a time-ordered world must itself be time-ordered. Main aim is to map progress in the real world on progress in the system that models it.
- The way of implementing the system is based on transformation of specification into efficient set of processes. These processes should be designed in such a manner that it would be possible to run them on available software and hardware

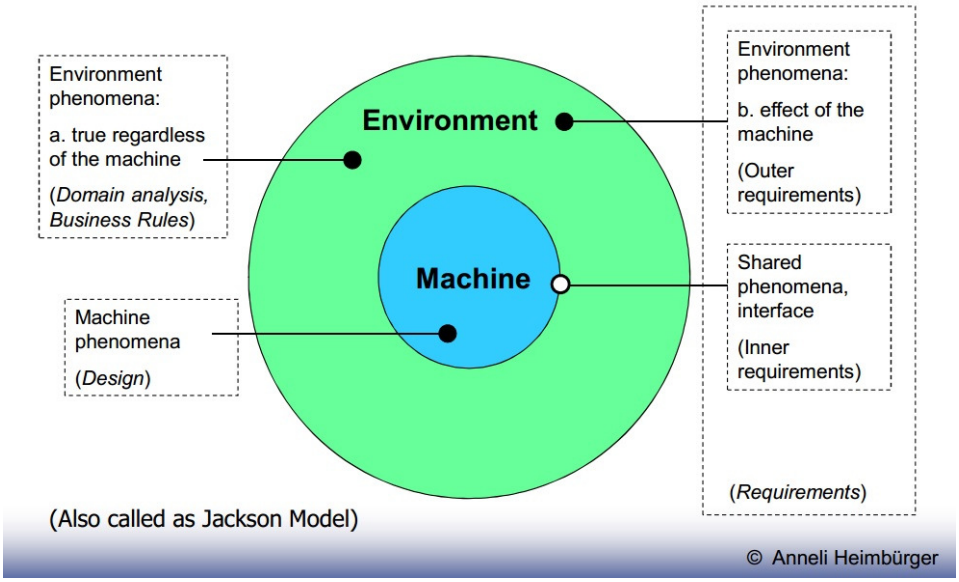


Fig: Machine Environment (Jackson) Model

Domain knowledge – description of environmental phenomena that occur regardless of the machine

- The average speed of manual spelling-check is about 5 pages per hour.
- Number of entries in Oxford English Dictionary is 252.200.

Outer requirements – described in terms of environmental phenomena, the desired effects in the environment occurring due to the machine influence

- A text shall be able to be spelling-checked effectively and efficiently

Inner requirements – description of shared phenomena, i.e. interaction of the machine and the environment at the machine-environment interface

- A computer-based system shall be developed able of checking spelling in electronic documents.
- If a word is found that is not in the system's dictionary, the user shall be given a possibility to select among similar correct words.

Design information – described in terms of machine phenomena, the internal structure and behavior of the machine

- The binary search algorithm (a binary search is an algorithm for locating the position of an element in a sorted list) is to be used

5. Starting with Jackson's machine-environment model, explain why RE is in fact a problem solving process (in addition to being problem-stating).

RE is mainly a problem-solving process as It is about answering the question "what machine's behavior on the boundary would achieve the desired effect on the environment".

Jackson's model is a method of system development that covers the software life cycle either directly or, by providing a framework into which more specialized techniques can fit. Jackson System Development can start from the stage in a project when there is only a general statement of requirements.

6. Describe all the different requirements engineering components.

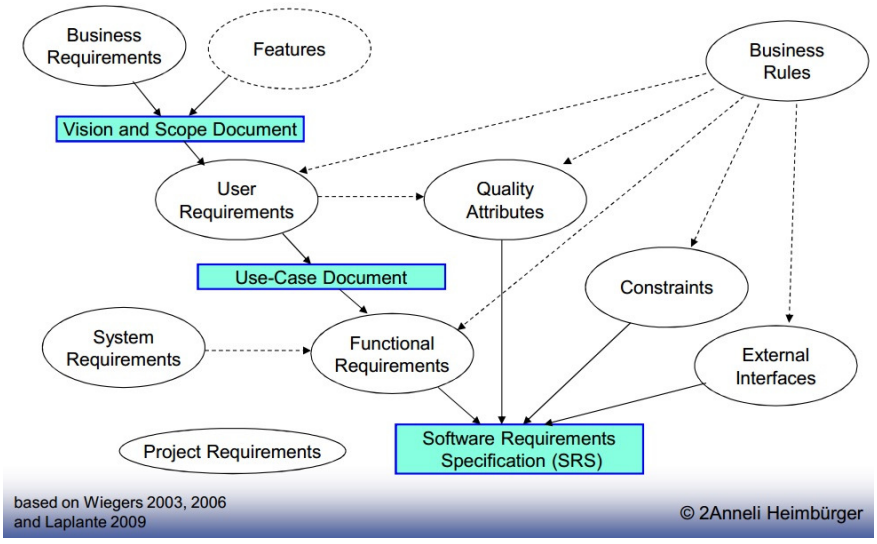
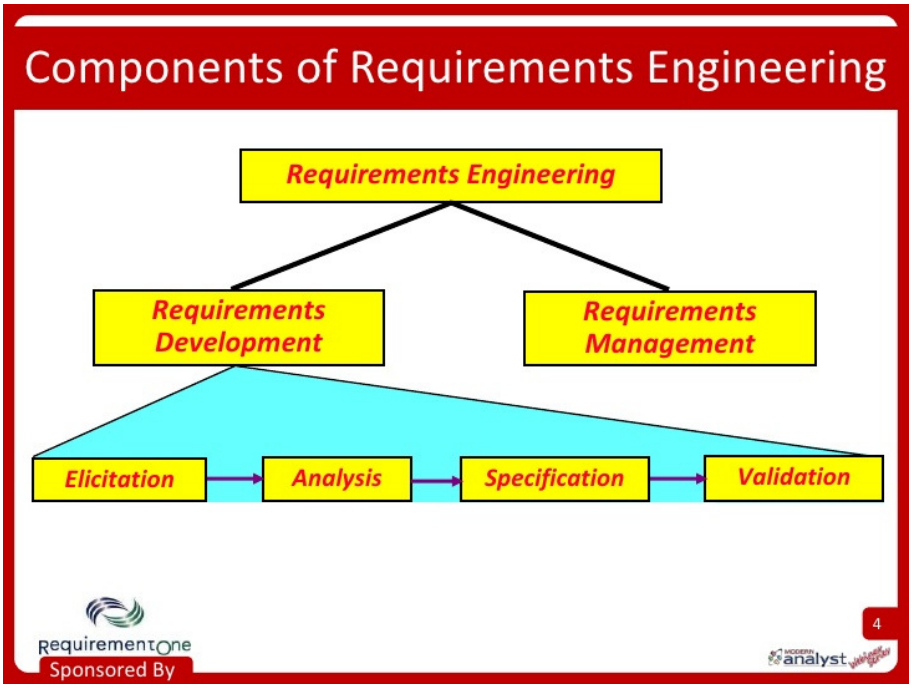


Fig: Requirement Engineering Component

- Business requirements: describe why the project is being undertaken.
- Features: the bulleted list on the product's sale package.

- User requirements: describe the tasks the users must be able to perform with the system.
- Functional requirements: specify the functionality the developers must build into the product to enable users to accomplish their tasks, thereby satisfying the business requirements.
- Non-functional requirements i.e. Quality Attributes: augment the description of the product functionality by describing the product's characteristics in various dimensions that are important to either users of developers
- Constraints: impose restrictions on the choices available to the developer for design and implementation of the system.
- External Interfaces: describe interfaces between the system and the external entities, with which the system must be able to communicate.
- Business Rules: include corporate policies, government regulations, industry standards, accounting practices, and computational algorithms.
    - Are not requirements, but affect requirements development. A part of domain knowledge.
- System Requirements: for software, the requirements for the whole computer-based system.
- Project Requirements: constraints placed on the development process of the system, e.g. budget, schedule, staff.

Requirements Engineering has two component requirements development and requirements management.



Requirement development is the part of the project during which the needs of the customer are gathered and translated into a specification of what the system must do. RD can further subdivide requirements development into elicitation, analysis, specification and validation.

- Elicitation: It is the process of gathering and documenting needs from stakeholders, identifying other requirements sources and applying techniques specified in the RMP to gather the information and document the needs

- Analysis: It is the process of analyzing the data gathered during elicitation, resolving conflicts, analyzing business rules, documenting assumptions, constraints and dependencies, and working with stakeholders to establish initial priorities.

- Specification: It is the process of defining functional and supplemental text based requirements and supporting them with various visualization techniques such as process models, UML diagrams, wireframes, white boarding etc

- Validation: It is the process of reviewing the requirement specifications and associated visualizations with the stakeholders for quality characteristics such as completeness, correctness, clarity, practicality, value etc.

Requirements management is the process of documenting, analyzing, tracing, prioritizing and agreeing on requirements and then controlling change and communicating to relevant stakeholders. It is a continuous process throughout a project.

7. What is system's quality attributes? Describe some of them.

System quality attributes are non-functional requirements used to evaluate the performance of a system. Some of system's qualities are:

- Availability: It is a measure of the planned up time during which the system is actually available for use and fully operational.
- Efficiency: It is a measure of how well the system utilizes processor capacity, disk space, memory or communication bandwidth.
- Flexibility: It measures how easy it is to add new capabilities to the product.
- Reliability: The probability of the software executing without failure for a specific period of time is known as reliability.
- Maintainability: It indicates how easy it is to correct a defect or modify the software. Maintainability depends on how easily the software can be understood, changed and tested.
- Portability: The effort required to migrate a piece of software from one operating environment to another is measure of portability.
- Testability: It is also known as verifiability, testability refers to the ease with which software components or the integrated product can be tested to look for defects

8. Discuss the place of requirements in the whole system engineering practice, according to different models, compare them.

Systems engineering is an interdisciplinary field of engineering that focuses on how to design and manage complex engineering systems over their life cycles. Issues such as reliability, logistics, coordination of different teams (requirements management), evaluation measurements, and other disciplines become more difficult when dealing with large or complex projects. Systems engineering deals with work-processes, optimization methods, and risk management tools in such projects. It overlaps technical and human-centered disciplines such as control engineering, industrial engineering, organizational studies, and project management. Systems Engineering ensures that all likely aspects of a project or system are considered, and integrated into a whole.

9. What is the difference between RD and RM? What activities constitute both?

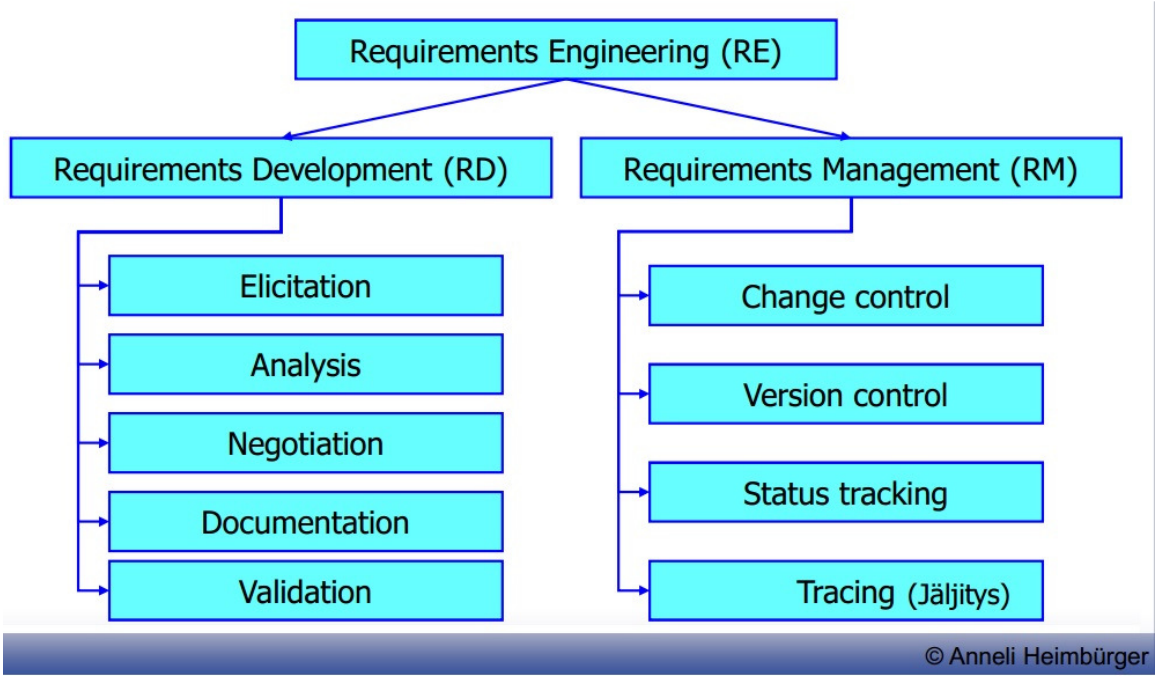


Fig: Requirements Engineering

Requirement Development establishes an agreement between the stakeholders (including the developing team) on the requirements of the system and it involves:

- Elicitation – the process of identifying system requirements from various sources through interviews, workshops, workflow and task analysis, document analysis, and other mechanisms.
- Validation (& verification) – the process of checking whether the requirements, as identified, do not contradict the expectations about the system of various stakeholders, and do not contradict each other.
- Analysis – the process of evaluating value/cost of different requirements, identifying dependencies between requirements, etc.
- Negotiation – the process of resolving conflicts between requirements, deciding which to accept, setting priorities.
- Documentation (specification) – the process of documenting requirements in a shareable and manageable form.

Whereas requirement management maintains the agreement and it Involves:

- Change control– managing changes to the requirements baseline, through reviewing proposed changes and evaluating the likely impact of each change before approving it, and incorporating approved changes into the project in a controlled way.
- Version control – managing document versions and requirements revisions.
- Requirements status tracking – defining a set of status values (Proposed, Approved, Implemented, Verified, Deleted, Rejected) for a requirement, and monitoring statuses throughout the project.
- Requirements tracing – managing dependency links between requirements, and tracing requirements up to their sources and down to corresponding design, source code, and test cases.
- The purpose of Requirement Development "RD" is to produce customer, product and product component requirements whereas Requirement Management "RM" is to manage the requirements of the project's products and product components.
- RD analyzes the customer, product and product component requirements whereas RM identifies inconsistencies between requirements and project's plans/work products.

- RD deals with communication between PM, domain and functional architect, designer and his team whereas RM deals with communication between requirements analysts and customer.
- RD is about the transformation of customer needs into requirements that can then evolve into design and/or code. This includes eliciting the customer needs(JAD sessions, interviews), transforming those needs into requirements, evolving them into product requirements, allocating the requirements across releases, teams, developers or modules, validating them, and ensuring that they fit within the customer constraints and assumptions whereas RM is all about maintaining the set of requirements that you have, and the process of accepting new ones.

10. Who is the requirements analyst? What skills this person should have, what are the specifics of her/his job?

The requirement analyst is the individual who has the primary responsibility to elicit, analyze, validate, specify, verify, and manage the real needs of the project stakeholders, including customers and end users. The requirement analyst serves as the conduit between the customer community and the software development team through which requirements flow.

Skills of requirement analyst are:

- Interviewing skills: to talk with individuals and groups about their needs and ask the right questions to surface essential requirements information

- Listening skills: to understand what people say and to detect what they might to hesitant to say

- Analytical skills: to critically evaluate the information gathered from multiple sources, reconcile conflicts, decompose high-level information into details, abstract up from low-level information to a more general understanding, distinguish

presented user requests from the underlying true needs, and distinguish solution ideas from requirements

- Facilitation skills: to lead requirements elicitation workshops

- Observational skills: to validate data obtained via other techniques and expose new areas from requirements

- Writing skills: to communicate information effectively to customers, marketing, managers and technical staff

- Organizational skills: to work with the vast array of information gathered during elicitation and analysis and to cope with rapidly changing information

- Interpersonal skills: to help negotiate priorities and to resolve conflicts among project stakeholders(such as customers, product management and engineering)

- Modelling skills: to represent requirements information in graphical forms that augment textual representations in natural language, including using modeling languages already established in the development organization

Job of requirement analyst are:

- Work with the project manager, product manager and project sponsor to document the product's vision and the project's scope.
- Identify project stakeholders and user classes. Document user class characteristics. Identify appropriate representatives for each user class and negotiate their responsibilities.
- Elicit requirements using interviews, document analysis, requirement workshop, storyboards, surveys, site visits.
- Write requirements specifications according to standard templates using natural language simply, clearly, unambiguously and concisely.

- Decompose high-level business and user requirements into functional requirements and quality
- Represent requirements using alternative views such as analysis models(diagrams), prototypes or scenarios where appropriate.

11. What do we mean when speaking of "requirements' quality"? What constitutes it? Explain some of important attributes of it.-

Requirements quality is constituted from different requirements attributes. These are classified as external and internal factors. External attributes are more important to users for example performance. Internal attributes indirectly contribute to customer satisfaction by making product easier to maintain e.g. Scalability .

12. What are the basic means of achieving the quality in general? How do they are used in RE?

Different projects demand different set of quality attributes to success. Common practical approach is to do following:

1. Begin by choosing rich set of quality attributes
2. Reduce attributes by thinking which attributes are likely to be important for project
3. Prioritize attributes. Attribute importance depends on project. Thats why one needs to compare chosen quality attributes what is more important.
4. Pin down concretely what these quality attributes mean and make sure customers understand what quality attributes mean
5. Specify quality requirements well. Too vague explanations aren't useful. Also make sure requirements are measurable. If requirement haven't been specified well enough its not good enough

13. Explain why completeness of requirements is a risky factor. How can we deal with it?

In practice no one can document every single requirement for any system. You always have some assumed or implied requirement there. There isn't any way to be certain that you have found all requirements of some system, but there are ways to find out that you have all requirements.

To elicit all requirements of system one can interview stakeholders, create workshops, observe users or do Questionnaires. These are only few way to ensure you have found all requirements. And no project team should expect to use only one technique

14. Explain the stakeholders' classification framework.

Stakeholders are people, group or organizations that are actively involved in project are affected by its outcome or are able to influence its outcome. Framework is for identify, manage and classify these stakeholders. Then for each identified stakeholder we detect its major value or benefits they will receive from the product, their likely attitudes toward the product, major features and characteristics of interest and any known constraints that must be accommodated. If stakeholder has none of these, they cannot be counted as project stakeholder.

15. Explain such concepts as "business requirements", "vision", "features", and "scope".

- "business requirements" specify primary benefits system will give to its sponsors, users and buyers. Business requirements directly influences what user requirements to implement.
- "vision" describes ultimate product that will achieve the business objectives. It describes what the product is about and what will it become.
- "features" system capability that provide value to user and its defined by functional requirements.
- "scope" the portion of the ultimate product vision that the current project will address. Draws boundary what is in and what is out for the current project.

16. Explain different levels and types of requirements.

Because there are so many different types of requirements information we need set of adjectives to overload term “requirement”
Software requirements include three distinct levels:

- Business requirement, a hig-level business objective of the organization that builds a product.
- User requirement, a goal or a task that specific class of users must be able to perform.
- Functional requirement, description of a behaviour that system wiill exhibit under specific conditions

17. What are project’s scope management and scope creep? What techniques we may use when defining and managing the scope?

Scope management is controlling new requirements of project. When new requirements appear they need to be approved.
Scope creep means uncontrolled changes of features in project. This can happen if project scope is poorly defined, underestimating complexity or lack of change control.

18. What is requirements elicitation? What techniques are available for that?

It is a process of identifying needs and constraints of the various stakeholders for a software project. Elicitation does not mean same thing as gathering. It is analytical process that includes activities to collect, discover extract and define requirements.

There are several elicitation techniques like:

- Interviews are the most simple way to elicit requirements. Just ask users.
- Workshops encourage stakeholders to define requirements together. Workshops may be time consuming and has to be well planned.

- Focus groups are representative group of users who gather to generate inpu and ideas on product functional and quality requirements.
- Observation, may sometimes be easier way to understand users job details than to ask them if the job is complex or users are too familiar of the job.
- Questionnaires are inexpensive way to survey large group of users to understand their needs across geographical boundaries
- System Interface analysis independent elicitation to examine systems that you system connect. This may reveal functional requirements regarding to exchange of data and services between systems.
- User Interface analysis studying existing UI systems to discover user and functional requirements.
- Document analysis is examining any existing documentation for potential software requirements

19. What did you learn?

Software requirements process is complicated ongoing process that requires constant managing and supervising. Requirements may easily grow too big and therefore ruin software project there are many practices to ensure requirement process succeeds.

20. Reflect your learning to your working experience/project experience or hobby related to software engineering. How can you apply your learning?

In real life software projects people who interact with customer are sometimes solely responsible for defining requirements. These requirements should be discussed with whole project team and with stakeholders. Also project team and stakeholders usually change during project lifecycle and because of that requirement process may change.

21. Explain the use case approach. Explain the elements of a use case description.

A use case describes a sequence of interactions between a system and an external actor. An actor is a person, another software system, or a hardware device that interacts

with the system to achieve a useful goal. Another name for actor is user role, because actors are roles that the members of one or more user classes can perform with respect to the system.

Use case approach is:

- The best-known and probably the best of known approaches to user requirements elicitation.
- Use cases are central in the Unified Process and UML.
- Is good because explicitly shifts the perspective from functional (what system must do) to user's (what users must be able to do with the system).
- Functional requirements are then developed based on the use cases.
- A use case describes a sequence of interactions between a system and an external actor(s).
- An actor is a person, another software system, or a hardware device that interacts with the system to achieve a useful goal.
    - Type of actors do not correspond to user classes. They rather represent the roles, which a user can play.
- One of the actors is an end-user

Element of use case:

Actor

An Actor models a type of role played by an entity that interacts with the subject (e.g., by exchanging signals and data), but which is external to the subject (i.e., in the sense that an instance of an actor is not a part of the instance of its corresponding subject). Actors may represent roles played by human users, external hardware, or other subjects. Note that an actor does not necessarily represent a specific physical entity but merely a particular facet (i.e., "role") of some entity that is relevant to the specification of its associated use cases. Thus, a single physical instance may play the role of several different actors and, conversely, a given actor may be played by multiple different instances.

Since an actor is external to the subject, it is typically defined in the same classifier or package that incorporates the subject classifier.

Association

An association specifies a semantic relationship that can occur between typed instances. It has at least two ends represented by properties, each of which is connected to the type of the end. More than one end of the association may have the same type.

An end property of an association that is owned by an end class or that is a navigable owned end of the association indicates that the association is navigable from the opposite ends; otherwise, the association is not navigable from the opposite ends.

Collaboration

A collaboration is represented as a kind of classifier and defines a set of cooperating entities to be played by instances (its roles), as well as a set of connectors that define communication paths between the participating instances. The cooperating entities are the properties of the collaboration.

A collaboration specifies a view (or projection) of a set of cooperating classifiers. It describes the required links between instances that play the roles of the collaboration, as well as the features required of the classifiers that specify the participating instances. Several collaborations may describe different projections of the same set of classifiers.

Constraint

A condition or restriction expressed in natural language text or in a machine readable language for the purpose of declaring some of the semantics of an element.

Dependency

A dependency is a relationship that signifies that a single or a set of model elements requires other model elements for their specification or implementation. This means that the complete semantics of the depending elements is either semantically or structurally dependent on the definition of the supplier element(s).

Extend

This relationship specifies that the behavior of a use case may be extended by the behavior of another (usually supplementary) use case. The extension takes place at one or more specific extension points defined in the extended use case. Note, however, that the extended use case is defined independently of the extending use case and is meaningful independently of the extending use case. On the other hand, the extending use case typically defines behavior that may not necessarily be meaningful by itself. Instead, the extending use case defines a set of modular behavior increments that augment an execution of the extended use case under specific conditions.

Generalization

A generalization is a taxonomic relationship between a more general classifier and a more specific classifier. Each instance of the specific classifier is also an indirect instance of the general classifier. Thus, the specific classifier inherits the features of the more general classifier.

Include

Include is a Directed Relationship between two use cases, implying that the behavior of the included use case is inserted into the behavior of the including use case. It is also a kind of Named Element so that it can have a name in the context of its owning use case. The including use case may only depend on the result (value) of the included use case. This value is obtained as a result of the execution of the included use case.

Note

A note (comment) gives the ability to attach various remarks to elements. A comment carries no semantic force, but may contain information that is useful to a modeler.

Realization

Realization is a specialized abstraction relationship between two sets of model elements, one representing a specification (the supplier) and the other represents an

implementation of the latter (the client). Realization can be used to model stepwise refinement, optimizations, transformations, templates, model synthesis, framework composition, etc.

System

If a subject (or system boundary) is displayed, the use case ellipse is visually located inside the system boundary rectangle. Note that this does not necessarily mean that the subject classifier owns the contained use cases, but merely that the use case applies to that classifier.

Use Case

A use case is the specification of a set of actions performed by a system, which yields an observable result that is, typically, of value for one or more actors or other stakeholders of the system.

22. Explain the user story approach in agile SD (Laplante's book).

User stories are one of the primary development artifacts for Scrum and Extreme Programming (XP) project teams. A user story is a very high-level definition of a requirement, containing just enough information so that the developers can produce a reasonable estimate of the effort to implement it.

User Stories provide a light-weight approach to managing requirements for a system. A short statement of function captured on an index card and/or in a tool. The details are figured out in future conversations between the team and the product owner or customers. This approach facilitates just in time requirements gathering, analysis and design by the following activities:

- Slicing user stories down in release planning
- Tasking user stories out in sprint planning
- Specifying acceptance test criteria for user stories early in development

User Stories are so good because it is:

- Understood equally well by everyone
- Useful for iteration planning
- Great for iterative development
- Encourage deferring of details
- Support opportunistic design
- Emphasize verbal communication

23.What are the potential problems and traps with the use case approach?

Potential problems with the use case approach:

- A set of Use Cases does not provide a system developer with all of the information that he needs about his client's needs, in order to produce a system that meets those needs. Use Cases are nutritionally deficient.
- Use Cases are extremely attractive. Much of the attraction is probably due to their simplicity. One doesn't have to work very hard to understand the basic Use Case concepts. With such a low-effort, no-sweat requirements-gathering technique available, it is tempting indeed to believe that all one has to do when gathering requirements is to create a list of Use Cases.
- The use of Use Cases, to the complete or virtual exclusion of other requirements-gathering techniques, has undesirable consequences, in the poor quality of the systems developed. But these consequences — the connection between requirements gathering and the eventual quality of the system as built — are largely invisible to both developers and developer management.

Traps with the use case approach

- Focusing on individual use cases and overlooking their interaction.
- CRUDL is a useful for quick analysis of correlation between use cases and various data entities.
- CRUDL stands for Create, Read, Update, Delete and List.

- A CRUDL matrix correlates system actions with data entities (individual data items or aggregates of data items) to make sure that you know where and how each data item is created, read, updated, deleted and listed.

24.What is requirements validation? How it is done?

Requirements validation is a feedback link. It communicates requirements, as constructed by the analysts, back to the stakeholders whose goals those requirements are supposed to meet, and to all those other stakeholders, with whose goals those requirements may conflict.

It is done by:

- Checking the right product is being built
- Ensuring that the software being developed (or changed) will satisfy its stakeholders
- Checking the software requirements specification against stakeholders goals and requirements

25. Explain what requirements review is. What is review reading technique and what other techniques we know?

Requirements review is a process in which a subset of the system stakeholders investigates the requirements represented in one or more of the available forms, using one or more of the available reading techniques, based on a subset of the quality criteria defined.

Requirement reading techniques provide very essential support to inspectors for filtration of ambiguous and doubted requirements from software requirements specifications (SRS). Inspectors use the reading techniques as a very effective strategy to simplify their workflow to purify the inconsistent requirements. There are many different reading techniques in literatures but practically ad-hoc reading and checklist-based are used in software industries.

Ad-hoc Reading Technique: This techniques works in ad-hoc mode it mean there is no proper mechanism in this technique as its name. All members in inspection team look for unwanted requirements from requirements specification without proper direction or guidelines. As such this technique is not so helpful for non experienced members. The defect correction is made on the bases of experience of inspectors.

Checklist based Reading: In this technique different questions based lists are provided to inspection team members and team members have to answer these questions which are related to consistency of requirements specifications. This reading technique is very helpful for inspectors to remove loop whole requirements specification by answering different question which they forgot during inspection process.

26. Explain why structured natural language stays the primary means of representing requirements.

Because all stakeholders must understand the requirements and natural language is flexible enough to document any possible requirement. Other methods are considered to be too heavy to use. Also natural language acts as common language between stakeholders.

27. Explain why and how politics becomes a significant part of RE process.

In complex projects requirements engineering is a largely political process. What goals different stakeholders pursue and who has power to make decisions.
Changes in political ecology may affect in functional requirements and continuous changes in thse may affect problems. Also functional decisions may change the balance of political ecology.
There are two types of politics:

- Functional politics - which of poblems deserve attention, whose interests will be served - concerns about the functional intent of the system

- Resource politics - to solving which of problems to allocate the available resources - concerns about the flow of resources going into the system

28.What is requirements negotiation, what are its goals, and the role of the requirements analyst?

Negotiation is the process of resolving conflicts between requirements, deciding which to accept, setting priorities. Its goals are to make conflicts explicit and to facilitate that right decisions are made. Right meaning that best solution is selected from group of solutions. Requirement analyst bridges communication between customer and developers.

29.What did you learn?

We learned that to succeed in requirement process, one needs to take account also political aspects. Communication between stakeholders is important and that is why we need common language. You can delegate negotiation to people whose job is to create bridge with stakeholders and developers which is important.

30. Reflect your learning to your working experience/project experience or hobby related to software engineering. How can you apply your learning

In real life software projects, customer usually changes a some of functional requirements during project. You can affect these changes by negotiating about them and not just accepting them. Also if customer is about to have some political ecology change it may be good thing to take account before going forward in project. It is also important to use some structured natural language that everyone understands what needs to be done, misunderstandings are usually reason why projects fail that's why discussion is important.