REQUIREMENTS ENGINEERING AND ANALYSIS

CORE CONCEPTS

Beatriz de Lucio

CORE CONCEPTS

WATERFALL VS. AGILE

1. Compare and contrast the requirements analysis and specification process in the waterfall software development model vs. the Agile software development model.

Traditional software development was done using the waterfall methodology, which spends a lot of time and effort on planning. By comparison, most modern software development is done using Agile software development methodologies including Scrum and lean frameworks which focus on quickly developing a potentially shippable/releasable increment (PSI).

In the analysis process, it will not be possible to perfectly satisfy the requirements of every stakeholder, and it is the software engineer's job to negotiate tradeoffs that are both acceptable to the principal stakeholders and within budgetary, technical, regulatory, and other constraints. A prerequisite for this is that all the stakeholders be identified, the nature of their "stake" analyzed, and their requirements elicited. This topic is concerned with the process of analyzing requirements to

- Detect and resolve conflicts between requirements.
- Discover the bounds of the software and how it must interact with its organizational and operational environment.
- Elaborate system requirements to derive software requirements.

For most engineering professions, the term "specification" refers to the assignment of numerical values or limits to a product's design goals. In software engineering, "software requirements specification" typically refers to the production of a document that can be systematically reviewed, evaluated, and approved. For complex systems, particularly those involving substantial non software components, as many as three different types of documents

are produced: system definition, system requirements, and software requirements. For simple software products, only the third of these is required.

Comparison of the requirements analysis and specification process in the waterfall model vs. the Agile model:

Software Development Model	
Waterfall	Agile
Linear	Incremental
Sequential	Iterative
Use Cases	User Stories
Phases	Sprints

FUNDAMENTAL ACTIVITIES OF SOFTWARE DEVELOPMENT

2. What are the (4-6) fundamental activities of software development (also stages of the Software Development Life Cycle) and how does each relate to requirements?

The fundamental activities of software development are:

1. Software **Requirements**
2. Software **Design**
3. Software **Construction**
4. Software **Testing**
5. Software **Maintenance**

Relationship between the fundamental activities of software development and requirements:

- Software Requirements focuses on elicitation, analysis, specification, and validation of the requirements, as well as the management of requirements during the whole life cycle of the software product.
- Software Design is the software engineering life cycle activity in which software requirements are analyzed in order to produce a description of the software's internal structure that will serve as the basis for its construction.
- Software Construction refers to the detailed creation of working software through a combination of coding, verification, unit testing, integration testing, and debugging.
- Software Testing consists of the dynamic verification that a program provides expected behaviors on a finite set of test cases, suitably selected from the usually infinite execution domain.
- The Software Maintenance efforts result in the delivery of a software product that satisfies user requirements.

FUNDAMENTAL REQUIREMENTS ACTIVITIES

3. (REQ.rfd) What are the (4-5) fundamental requirements activities and how are they performed? (In other words, what is done during the specification/requirements stage of the software development life cycle?) (Within your answer, define a requirement and identify requirements characteristics.)

Fundamental requirements activities and how are they performed:

1. **Elicitation**: concerned with the origins of software requirements and how the software engineer can collect them. It is the first stage in building an understanding of the problem the software is required to solve. It is fundamentally a human activity and is where the stakeholders are identified, and relationships established between the development team and the customer.
2. **Analysis**: a prerequisite for this is that all the stakeholders be identified, the nature of their "stake" analyzed, and their requirements elicited. It will not be possible to perfectly satisfy the requirements of every stakeholder, and it is the software engineer's job to negotiate tradeoffs that are both acceptable to the principal stakeholders and within budgetary, technical, regulatory, and other constraints.
3. **Specification**: relates to the production of a document that can be systematically reviewed, evaluated, and approved.
4. **Validation**: is the process by which engineers ensure that the system will meet these needs and requirements Validation is used to ensure that one is working the right problem.

A software **requirement** is a property that must be exhibited by something to solve some problem in the real world.

Requirements characteristics:

- **Necessary** The requirement defines an essential capability, characteristic, constraint, and/or quality factor. If it is not included in the set of requirements, a deficiency in capability or characteristic will exist, which cannot be fulfilled by implementing other requirements.
- **Appropriate** The specific intent and amount of detail of the requirement is appropriate to the level of the entity to which it refers (level of abstraction). This includes avoiding unnecessary constraints on the architecture or design to help ensure implementation independence to the extent possible.
- **Unambiguous** The requirement is stated in such a way so that it can be interpreted in only one way.
- **Complete** The requirement sufficiently describes the necessary capability, characteristic, constraint, or quality factor to meet the entity need without needing other information to understand the requirement.
- **Singular** The requirement should state a single capability, characteristic, constraint, or quality factor.
- **Feasible** The requirement can be realized within entity constraints (e.g., cost, schedule, technical, legal, regulatory) with acceptable risk.
- **Verifiable** The requirement is structured and worded such that its realization can be proven (verified) to the customer's satisfaction at the level the requirements exist.
- **Correct** The requirement must be an accurate representation of the entity need from which it was transformed.

- **Conforming** The individual requirements should conform to an approved standard template and style for writing requirements, when applicable.

ELICITING REQUIREMENTS

4. (REQ.er) How would you go about eliciting requirements for a software project?

Eliciting requirements is concerned with the origins of software requirements and how the software engineer can collect them. It is the first stage in building an understanding of the problem the software is required to solve. It is fundamentally a human activity and is where the stakeholders are identified, and relationships established between the development team and the customer.

General Techniques:

1. **Interviews.** Interviewing stakeholders is a "traditional" means of eliciting requirements. It is important to understand the advantages and limitations of interviews and how they should be conducted.
2. **Survey or Questionnaire**: used to identify organizational gaps or stakeholder concerns.
3. **Scenarios**. Scenarios provide a valuable means for providing context to the elicitation of user requirements. They allow the software engineer to provide a framework for questions about user tasks by permitting "what if" and "how is this done" questions to be asked. The most common type of scenario is the use case description.
4. **Prototypes**. This technique is a valuable tool for clarifying ambiguous requirements. They can act in a similar way to scenarios by providing users with a context within which they can better understand what information they need to provide. There is a wide range of prototyping techniques—from paper mockups of screen designs to beta-test versions of software products—and a strong overlap of their separate uses for requirements elicitation and for requirements validation. Low fidelity prototypes are often preferred to avoid stakeholder "anchoring" on minor, incidental characteristics of a higher quality prototype that can limit design flexibility in unintended ways.
5. **Facilitated meetings**. The purpose of these meetings is to try to achieve a summative effect, whereby a group of people can bring more insight into their software requirements than by working individually. They can brainstorm and refine ideas that may be difficult to bring to the surface using interviews. Another advantage is that conflicting requirements surface early on in a way that lets the stakeholders recognize where these occur.
6. **Observation**. The importance of software context within the organizational environment has led to the adaptation of observational techniques such as ethnography for requirements elicitation. Software engineers learn about user tasks by immersing themselves in the environment and observing how users perform their tasks by interacting with each other and with software tools and other resources.
7. **User stories**. This technique is commonly used in adaptive methods and refers to short, high level descriptions of required functionality expressed in customer terms. A typical user story has the form: "As a <role>, I want <goal/desire> so that <benefit>." A user story is intended to contain just enough information so that the developers can produce a reasonable estimate of the effort to implement it.

8. **Other techniques.** A range of other techniques for supporting the elicitation of requirements information exist and range from analyzing competitors' products to applying data mining techniques to using sources of domain knowledge or customer request databases.

DOCUMENTS AND DIAGRAMS

5. (REQ.rsd) What are some documents and diagrams you would create or reference during the requirements engineering process? (include information about order of creation / reference, target audience, structure, key contents, and benefits)

1. **Business Requirements Specification (BRS)**
    a. **Purpose**: describes the organization's motivation for why the system is being developed or changed, defines processes and policies/rules under which the system is used and documents the top-level requirements from the stakeholder perspective including expressing needs of users/operators/maintainers as derived from the context of use in a specific, precise, and unambiguous manner.
    b. **Key Content**: at the organization level, the organizational environment, goals and objectives, the business model, and the information environment, and, at the business operation level, the business operation model, business operation modes, business operational quality, organizational formation, and concept of the proposed system. Typical types of requirements included in the BRS are organizational requirements and business requirements.
    c. **Created By**: the business itself, often with a business analyst
    d. **Target Audience**:
        i. A business analyst or representative user from the business to review and discuss the business model or operation
        ii. Business management to verify and revise
        iii. A system analyst to review and discuss potential technical solutions
        iv. Software/systems engineers to create the SyRS and/or SRS
2. **Stakeholder Requirements Specification (StRS)**
    a. **Purpose**: describes the organization's motivation for why the system is being developed or changed, defines processes and policies/rules under which the system is used and documents the top-level requirements from the stakeholders' perspective including expressing needs of users/operators/maintainers as derived from the context of use in a specific, precise, and unambiguous manner.
    b. **Key Content**: Typical types of stakeholder requirements included in the StRS are organizational requirements, business requirements and user requirements. There is no one optimal organization for all projects.
    c. **Created By**: should be specified by the stakeholders. The stakeholders should be responsible for the content of the specification. Could be made by a business analyst or requirements engineer.
    d. **Target Audience**:
        i. Stakeholders to review and achieve consensus

   ii. Software / systems engineers to create the SyRS and/or SRS

3. **System Requirements Specification (SyRS)**
   a. **Purpose**: identifies the technical requirements for the selected system-of-interest and usability for the envisaged human-system interaction. It defines the high-level system requirements from the domain perspective, along with background information about the overall objectives for the system, its target environment and a statement of the constraints, assumptions, and non-functional requirements. Provide a description of what the system should do, in terms of the system's interactions or interfaces with its external environment.
   b. **Key Content**: completely describe all inputs, outputs and required relationships between inputs and outputs. Can be a paper document, models, prototypes, other non-paper document representations or any combination.
   c. **Created By**: a systems engineer or software engineer
   d. **Target Audience**:

      i. The technical community who will specify and build the system.
      ii. Needs to be understandable by both the acquirer and the technical community.

4. **Software Requirements Specification (SRS)**
   a. **Purpose**: a specification for a particular software product, program, or set of programs that performs certain functions in a specific environment.
   b. **Key Content**: all of the required capabilities of the specified software product to which it applies, as well as documenting the conditions and constraints under which the software has to perform, and the intended verification approaches for the requirements.
   c. **Created By**: usually a software engineer but could be a technical writer, a systems architect, or a software programmer; one or more representatives of the supplier, one or more representatives of the acquirer, or by both.
   d. **Target Audience**:

      i. stakeholders to review and validate
      ii. the development team and team managers as a basis for coding
      iii. quality assurance / testers as a basis for designing tests
      iv. operations
      v. maintenance
      vi. project consultants

VALIDATE REQUIREMENTS

6. (REQ.rv) How do you validate requirements?

Validation is used to ensure that one is working on the right problem.

Techniques:

1. **Reviews and inspections**: A group of reviewers is assigned a brief to look for errors, mistaken assumptions, lack of clarity, and deviation from standard practice. The composition of the group that conducts the review is important (at least one

representative of the customer should be included for a customer-driven project, for example), and it may help to provide guidance on what to look for in the form of checklists. Reviews may be constituted on completion of the system definition document, the system specification document, the software requirements specification document, the baseline specification for a new release, or at any other step in the process.

2. **Prototyping to validate requirements**: An executable model of the system is demonstrated to the customer and end users to validate and ensure if it meets their needs. Prototyping is usually used when the requirements are not clear. So, we make a quick design of the system to validate the requirements. If it fails, we then refine it, and check again, until it meets the customer needs. This will decrease the cost as a result of having a clear, understandable, consistent requirements.
3. **Acceptance test design**: An essential property of a software requirement is that it should be possible to validate that the finished product satisfies it. Requirements that cannot be validated are just "wishes." An important task is therefore planning how to verify each requirement. In most cases, designing acceptance tests does this for how end-users typically conduct business using the system. To be validated, they must first be analyzed and decomposed to the point where they can be expressed quantitatively.
4. **Validating product quality attributes**: Validating availability, correctness, efficiency, expandability, flexibility, interoperability, maintain-ability, portability, reliability, reusability, supportability, survivability, and usability.
5. **Requirements interaction analysis** (e.g., feature interaction): Focuses on the communications or control flow relations between entities used to accomplish a specific task or function within the software model. This analysis examines the dynamic behavior of the interactions between different portions of the software model, including other software layers (such as the operating system, middleware, and applications). It may also be important for some software applications to examine interactions between the computer software application and the user interface software. Some software modeling environments provide simulation facilities to study aspects of the dynamic behavior of modeled software. Stepping through the simulation provides an analysis option for the software engineer to review the interaction design and verify that the different parts of the software work together to provide the intended functions.
6. **Formal requirements analysis**: The formal expression of requirements requires a language with formally defined semantics. it enables requirements expressed in the language to be specified precisely and unambiguously. requirements can be reasoned over, permitting desired properties of the specified software to be proven. Most formal analysis is focused on relatively late stages of requirements analysis.

CRITICAL THINKING

7. (FGCUScholars 2) What is critical thinking? How do you think critically?

There are many definitions of critical thinking. Read a few. Memorize at least one that makes sense to you.

- Evaluate and analyze.
- Thinking about thinking to improve thinking.
- Recognize assumptions, evaluate arguments, draw conclusions.
- "Critical thinking is the intellectually disciplined process of actively and skillfully conceptualizing, applying, analyzing, synthesizing, and/or evaluating information gathered from, or generated by, observation, experience, reflection, reasoning, or communication, as a guide to belief and action." - criticalthinking.org

Techniques:

- Going through the Elder Paul model's **Elements of Thought** wheel
- **5 Why's**: iterative interrogative technique used to explore the cause-and-effect relationships underlying a particular problem.
- **SEE-I**: State, Elaborate, Exemplify, and Illustrate. Clarification and understanding
- Evaluate an Argument with Just ONE **Flowchart**

Is generally useful to:

- **Listen** more than I talk (2 ears, 1 mouth).
- **Avoid quick** decisions. Get all the **facts** first.
- "I would need to learn more before forming an opinion." is a great thing to say.

Concepts:

- Bias - "inclination or prejudice for or against one person or group, especially in a way considered to be unfair"
- Agenda - "underlying intentions or motives of a particular person or group"
- Arguments - "In everyday life, people often use "argument" to mean a quarrel between people. But in logic and critical thinking, an argument is a list of statements, one of which is the conclusion, and the others are the premises or assumptions of the argument."
- Objectivity - "lack of favoritism toward one side or another: freedom from bias"
- Bloom's Taxonomy and Lower Order Thinking Skills (LOTS) vs Higher Order Thinking Skills (HOTS)