**Chapter 4 Notes**

**Chapter 4**

**User Requirements**: Statements, in a natural language plus diagrams, of what services the system is expected to provide to system users and the constraints under which it must operate

* These tend to be general
* Readers are NOT usually concerned with how the system will be implemented

**System Requirements**: More detailed descriptions of the SW system’s functions, services, and operational constraints. This document (aka functional specification) should define exactly what is to be implemented.

* These tend to be more specific
* Readers need to know more precisely what the system will do

**Functional and Non-Functional Requirement (Section 4.1)**

1. System Requirements (not User) are classified either as Functional or Non-Functional
2. Functional:
   1. Statements of services the system should provide
   2. How the system should react to particular inputs
   3. How the system should behave in particular situation
   4. Maybe, state what the system should NOT do
3. Non-Functional:
   1. Constraints on the services or functions offered by the system
   2. Timing constraints
   3. Constraints on development process
   4. Constraints imposed by standards

**Functional Requirements (Section 4.1.1)**

1. Describe what the system should do
2. Define specific facilities to be provided by the system (these have been taken from the User Requirements document)
3. Should be COMPLETE and CONSISTENT
   1. Completeness: All services required by the user should be define
   2. Consistency: Requirements should not have contradictory definitions

**Non-Functional Requirements (Section 4.1.2)**

1. Requirements that are not directly concerned with the specific services delivered by the system to its users
2. Usually deal with performance, security, availability, reliability, response time, and store occupancy requirements/constraints.
3. It is difficult to relate components to Non-functional requirements (it is easier to relate component to functional requirements) because:
   1. Non-functional requirements may affect the overall architecture of a system rather than individual components (i.e. Performance)
   2. A single non-functional component, such as security requirement, may generate a number of related functional requirements
4. Types of Non-Functional Requirements
   1. Product Requirements
      1. Specify or constrain the behavior of the software (Ex. Performance requirements, reliability requirements, security requirements, and usability requirements)
   2. Organizational Requirements
      1. Broad system requirements derived from policies and procedures in the customer’s and developer’s organization. (Ex. Operational process requirements, development process requirements, and environmental process requirements)
   3. External Requirements
      1. All requirements that are derived from factors external to the system and its development process (Ex. Regulatory requirements, legislative requirements, and ethical requirements)
5. Metrics for Specifying Non-Functional Requirements
   1. You should write Non-Functional Requirements quantitatively so that they can objectively tested
   2. Metrics
      1. Speed
         1. Processed transactions/second, Response time, Screen refresh time
      2. Size
         1. Mbytes, # of ROM chips
      3. Ease of Use
         1. Training time, Number of help frames
      4. Reliability
         1. Mean time to failure, Probability of unavailability, rate of failure occurrence, and availability
      5. Robustness
         1. Time to restart after failure, % of events causing failure, probability of data corruption on failure
      6. Portability
         1. % of target dependent statements, # of target systems
   3. Non-Functional requirements often conflict with other Non-Functional and Functional Requirements.

**The SW Requirements Document (Section 4.2)**

1. AKA SRS (SW Requirements Specification)
   1. Includes both the User Requirements and Detailed Specification of the System Requirements
      1. Agile Methods argue that the moment SRS is compiled, it is quickly out of date so there is no point to do it at all
2. Users of a Requirements Document
   1. System Customers
      1. Specify the requirements check that they meet their need. Specify changes to the requirements
   2. Managers
      1. Use the requirements document to plan a bid for the system to plan the system development process
   3. System Engineers
      1. Use the requirements to understand what system is to be developed
   4. System Test Engineers
      1. Use the requirements to develop validation tests
   5. System Maintenance Engineers
      1. Use the requirements to understand the system and the relationships between components
3. Structure of Requirements Document
   1. Preface
      1. Define the readership of the document, version history of the document, and summary of changes of each version
   2. Introduction
      1. Describe the need for the system. Describe the function, and explain how it will work with other systems.
   3. Glossary
      1. Define technical terms
   4. User Requirement Definition
      1. Describe the services provided for the user. Non-functional Requirements should be described here. Product and process standards that must be followed should be specified
   5. System Architecture
      1. High-level overview of the anticipated system architecture
   6. System Requirements Specification
      1. Describe Functional and Non-Functional requirements in more detail
   7. System Models
      1. Graphical models showing relationships between system components, the system, and its environment.
   8. System Evolution
      1. Fundamental assumptions on which the system is based, and any anticipated changes due to hardware evolution, changing user needs, and so on.
   9. Appendices
      1. Specific information related to the application (ex. Hardware and Database specifications)
   10. Index

**Requirements Specification (Section 4.3)**

1. Is the process of writing down the User and System Requirements in a requirements document
2. User Requirements should describe the Functional and Non-Functional Requirements sot that they are understandable by system users who don’t have detailed technological knowledge.
3. System Requirements are expanded version of the User Requirements that are used by SW Engineers as the starting point for the system design.
   1. They should not be concerned with how the system should be designed or implemented
   2. It is practically impossible to exclude all design information because:
      1. You may have to design an initial architecture of the system to help structure the requirements specification
      2. System must interoperate with existing systems, which constrain the design and impose requirements on the new system
      3. The use of specific architecture to satisfy Non-Functional Requirements may be necessary.
4. Writing Requirements Specifications
   1. User
      1. Written in natural language with appropriate diagrams and tables
   2. System
      1. Natural Language sentences
         1. Written using numbered sentences each expressing one requirement
      2. Structured natural language
         1. Written in natural language on a standard form or template. Each field provides info about an aspect of the requirement
      3. Design description language
         1. Uses language like a programming language but with more abstract features to specify the requirements by defining an operational model of the system.
      4. Graphical notations
         1. Graphical models like UML
      5. Mathematical specifications
         1. Based on mathematical concepts such as finite state machines

**Natural Language Specification (Section 4.3.1)**

1. Guidelines
   1. Invent a standard format and ensure that all requirement definitions adhere to it
   2. Use language consistently to distinguish between mandatory and desirable requirements
   3. Use text highlighting to pick out key parts of the requirement
   4. Do not assume that readers understand technical SWE language
   5. Try to associate a rationale with each user requirement

**Structured Specifications (Section 4.3.2)**

1. The freedom of the requirements writer is limited and all requirements are written in a standard way
2. Use templates to specify system requirements and may use programming language constructs to show alternatives, iteration and may highlight key elements using shading or different fonts
3. When a standard form is used for requirements, include the following info:
   1. Description of the function or entity being specified
   2. Description of its inputs and where these come from
   3. Description of its outputs and where these go to
   4. Info about the info that is needed for the computation or other entities in the system that are used
   5. A description of the action to be taken
   6. If a functional approach is used, a pre-condition and a post-condition
   7. Description of the side effects (if any) of the operation

**Requirements Engineering Processes (Section 4.4)**

1. Requirements Engineering 4 Phases
   1. Feasibility Study
      1. Is system useful to the business
   2. Elicitation and Analysis
      1. Discovering requirements
   3. Specification
      1. Converting these requirements into some standard form
   4. Validation
      1. Checking that the requirements actually define the system that the customer wants
2. Requirements engineering is an iterative process in which the activities are interleaved in a spiral…the output is the System Requirements Document.
   1. This spiral accommodates approaches to development where the requirements are developed to different levels of detail
   2. The # of iterations around the spiral can vary so it can be exited after some or all of the user requirements have been elicited
   3. Agile development can be used so that the requirements and the system implementation are developed together

**Requirements Elicitation and Analysis (Section 4.5)**

1. This phases follows the Feasibility Study phase
2. In this phase, SW Engineers work with customer and system end-users to find out about the application domain, what services the system should provide, the required performance of the system, hardware constraints, etc.
3. Process
   1. 1) Requirements Discovery
      1. Interacting with stakeholders of the system to discover their requirements
   2. 2) Requirement Classification and Organization
      1. Takes the unstructured collection of requirements, groups related requirements, and organizes them into coherent clusters (easiest way to do this is to associate requirements with system components)
   3. 3) Requirements Prioritization and Negotiation
      1. Concerned with prioritizing requirements and finding and resolving requirements conflicts through negotiation.
   4. 4) Requirements Specification
      1. Requirements are documented and input into the next round of the spiral
4. Difficulty with eliciting and understanding requirements from stakeholders:
   1. They often don’t know what they want from a computer system except in the most general terms…they may find it difficult to articulate what they want the system to do…they may make unrealistic demands because they don’t know what is and isn’t feasible
   2. They naturally express requirements in their own terms and with implicit knowledge of their own work…engineers may not understand the stakeholders domain
   3. Different stakeholders have different requirements and they may express these in different ways…engineers must find all commonalities and conflicts
   4. Political factors may influence the requirements of a system
   5. Economic an business environment in which the analysis takes place is dynamic

**Requirements Discovery (Section 4.5.1)**

1. Is the process of gathering information about the required system and existing system, and distilling the user and system requirements from this information
   1. Sources of info may be documentation, system stakeholders, and specifications of similar systems
2. Also known as Requirements Elicitation

**Interviewing (Section 4.5.2)**

1. Formal/Informal interviews with system stakeholders are part of most requirements engineering processes
2. The requirements engineering team outs questions to stakeholders about the system they they currently use and the system to be developed. Requirements are derived from the answers to these questions.
3. 2 Types of Interviews
   1. Closed Interviews
      1. Stakeholders answer pre-defined questions
   2. Open interviews
      1. No pre-define agenda
4. It can be difficult to elicit domain knowledge through interview because:
   1. All application specialists use terminology and jargon that are specific to a domain
   2. Some domain knowledge is so familiar to stakeholders that they either find it difficult to explain or they think it is so fundamental that it isn’t worth mentioning
      1. Ex. For a librarian, it goes without saying that all acquisitions are cataloged before they are added to the library. However, this may not be obvious to the interviewer, and so it isn’t taken into account in the requirements
5. Effective interviewers have 2 characteristics:
   1. They are open-minded, avoid pre-conceived ideas about the requirements, and are willing to listen to stakeholders. If the stakeholder comes up with surprising requirements, then they are willing to change their mind about the system.
   2. They prompt the interviewee to get discussions going using a springboard question, a requirements proposal, or by working together on a prototype system.

**Scenarios (Section 4.5.3)**

1. People find it easier to relate to real-life examples rather than abstract descriptions…engineers can use information gained here for requirements
2. They are descriptions of example interaction sessions
3. Scenario starts with an outline of the interaction. It may include
   1. Description of what the system and users expects when the scenario starts
   2. Description of the normal flow of events in the scenario
   3. Description of what can go wrong and how this is handled
   4. Info about other activities that might be going on at the same time
   5. Description of the system state when the scenario finishes

**Use Cases (Section 4.5.4)**

1. Are a requirements discovery technique
2. They identify the actors involved in an interaction and names the type of interaction. Information describing the interaction with the system is also included.
3. Some consider use cases and scenarios to be the same
4. Use Cases identify the individual interactions between the system and its users or other systems.

**Ethnography (Section 4.5.5)**

1. Motive: SW systems are used in social and organizational context and software requirements may be derived or constrained by that context. Satisfying these social and organizational requirements is often critical for the success of the system.
2. Is an observational technique that can be used to understand operational processes and help derive support requirements for these processes
   1. An analyst is placed in the working environment where the system will be used. The day-to-day work is observed and notes made of the actual tasks in which participants are involved.
      1. This helps see the actual ways that people work and discover implicit system requirements
3. Stages in Ethnography and Prototyping for Requirements Analysis
   1. Ethnographic Analysis
   2. Debriefing Meetings
   3. Focused Ethnography
   4. Prototype Evaluation
   5. System Prototyping
   6. Generic System Development
4. It is effective for discovering 2 types of requirements
   1. Requirements that are derived from the way in which people actually work rather than the way in which process definitions say they ought to work
   2. Requirements that are derived from cooperation and awareness of other people’s activities
5. Ethnography can be combined with prototyping (as shown in bullet #3 above)

**Requirements Validation (Section 4.6)**

1. Is the process of checking that requirements actually define the system that the customer really wants.
2. Process
   1. Validity Checks
      1. A user may think that a system is needed to perform certain functions. However, further thought and analysis may identify additional or different functions that are required.
   2. Consistency
      1. Requirements in the document should not conflict
   3. Completeness checks
      1. The requirements document should include requirements that define all functions and the constraints intended by the system user
   4. Realism Checks
      1. Using knowledge of existing technology, the requirements should be checked to ensure that they can actually be implemented.
   5. Verifiability
      1. To reduce the potential for dispute between customer and contractor, system requirements should always be written so that they are verifiable (i.e. have tests that demonstrate system complies)
3. Requirements Validation techniques
   1. Requirements reviews
      1. Requirements are analyzed systematically by a team of reviewers who check for errors and inconsistencies
   2. Prototyping
      1. An executable model of the system in question is demonstrated to end-users and customers. They can experiment with this model to see if the meets their real needs
   3. Test-case generation
      1. Requirements should be testable

**Requirement Management/Evolution (Section 4.7)**

1. Requirements for a large system are always changing
2. Requirements Evolution
   1. Initial Understanding of Problem
   2. Initial Requirements
   3. Changed Understanding of Problem
   4. Changed Requirements
3. Why is change inevitable?
   1. Business and technical environment of the system always changes after installation. New hardware may be introduced, it may be necessary to interface the system with other systems, business priorities may change, and new legislations and regulations may be introduced.
   2. People who pay for a system and the users of that system are rarely the same people.
   3. Large systems usually have a diverse user community, with many users having different requirements and priorities that may be conflicting or contradictory
4. Is the process of understanding and controlling changes to system requirements.

**Requirements Management Planning (Section 4.7.1)**

1. 1st STAGE of REQUIREMENTS MANAGEMENT PROCESS
2. Establishes the level of requirements management detail that is required
3. During the requirements management stage you must decide on:
   1. Requirements identification
      1. Each requirement must be uniquely identified
   2. A change management process
      1. The set of activities that assess the impact and cost of changes
   3. Traceability policies
      1. Policies that define the relationships between each requirement and between the requirements and the system design that should be recorded.
   4. Tool Support
      1. Tools that may be used range from specialist requirements management systems to spreadsheets and simple database systems
4. You’ll need tool support for:
   1. Requirements change
      1. Requirements should be maintained in a secure, managed data store that is accessible to everyone involved in the requirements engineering process
   2. Change management
      1. Process of change management is simplified if active tool support is available
   3. Traceability management
      1. Tool support for traceability allows related requirements to be discovered.

**Requirements Change Management (Section 4.7.2)**

1. Should be applied to all proposed changes to a system’s requirements after the requirements document has been approved.
2. Process
   1. Problem Analysis and Change Specification
      1. The problem or change proposal is analyzed to check that it is valid. This analysis is fed back to the change requestor who may respond with a more specific requirements change proposal, or decide to withdraw the proposal
   2. Change Analysis and Costing
      1. The effect of the proposed change is assessed using traceability information and general knowledge of the system requirements. The cost of making the change is estimated both in terms of modifications to the requirements document and, if appropriate, to the system design and implementation
   3. Change Implementation
      1. The requirements documents and, where necessary, the system design and implementation are modified.

**Key Points**

1. Requirements for a software system set out what the system should do and define constraints on its operation and implementation
2. Function requirements are statements of the services that the system must provide or are descriptions of how some computations must be carried out
3. Non-functional requirements often constrain the system being developed and the development process being used. These might be product requirements, organizational requirements, or external requirements.
4. The SW requirements document is an agreed statement of the system requirements.
5. The requirements engineering process includes a feasibility study, requirements elicitation and analysis, requirements specification, requirements validation, and requirements management.
6. Requirements elicitation and analysis, is an iterative process that can be represented as a spiral of activities – requirements discovery, requirements classification and organization, requirements negotiation, and requirements documentation.
7. Requirements validation is the process of checking the requirements for validity, consistency, completeness, realism, and verifiability.
8. Business, organizational, and technical changes inevitably lead to changes to the requirements for a SW system.