CS 490 – Requirements Engineering

Instructor: Yongjie Zheng

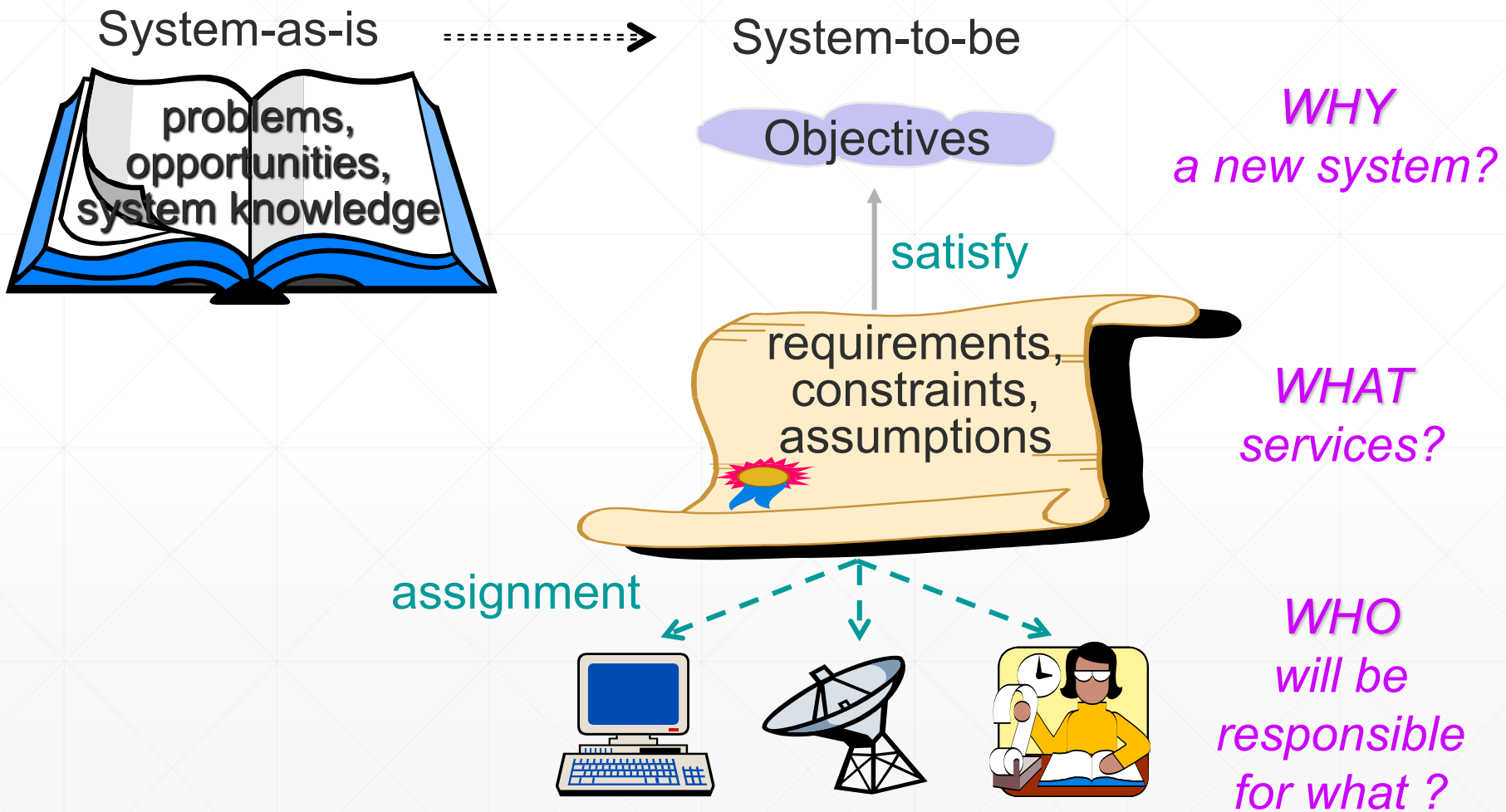
*The hardest single part of building a software system is **deciding precisely what to build***. No other part of the conceptual work is so difficult as **establishing the detailed technical requirements, including all the interfaces to people, to machines, and to other software systems**. No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later.

- Frederick P. Brooks Jr.

Definition of Requirements Engineering (RE)

- RE is a coordinated set of activities for exploring, evaluating, documenting, consolidating, revising, and adapting the **objectives**, **capabilities**, **qualities**, **constraints**, and **assumptions** that the **system–to-be** should meet based on problems raised by the system-as-is and opportunities provided by new technologies.

The Scope of RE: the *WHY*, *WHAT*, *WHO* Dimensions



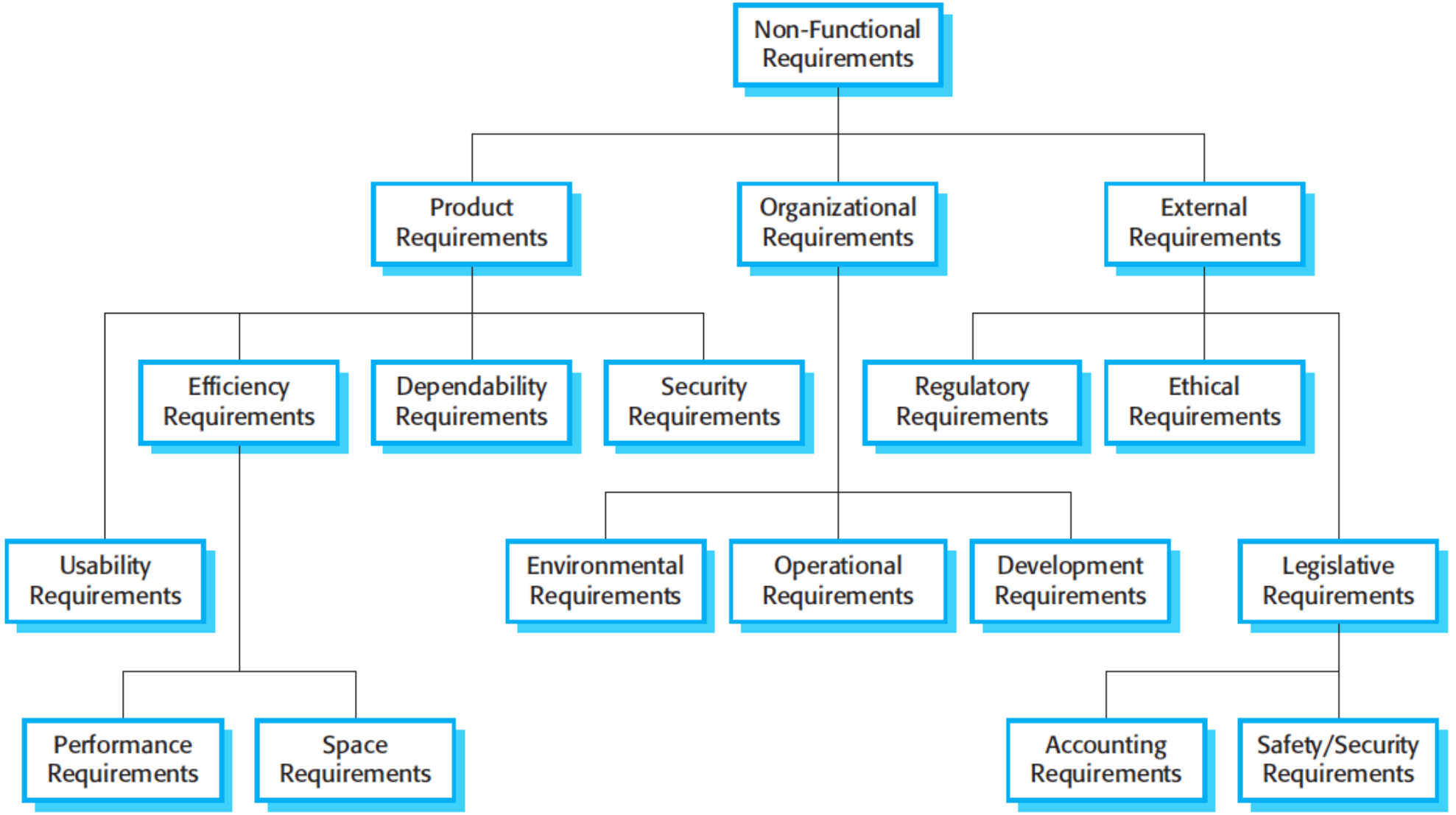
Why, What, and Who of RE

- Why --- Identify, analyze, refine the system-to-be's **objectives.**
- What --- Identify & define the system-to-be's **functional services** (software services, associated manual procedures).
  - to satisfy the identified objectives
  - according to quality **constraints**: security, performance, ...
  - based on realistic **assumptions** about the environment
- Who --- assign **responsibilities** for the objectives, services, constraints among system-to-be components, including **people**, **devices**, **pre-existing software**, and **software-to-be**.

Functional and Non-functional Requirements

- **Functional requirements** (aka **features**) describe **what** the system should do.
- **Non-functional requirements** describe **how** the system should provide services. (e.g., constraints on the services or functions offered by the system).
  - **Quality requirements**: performance, security, reliability, extensibility, robustness, etc.
  - **Compliance requirements**: national laws, international regulations, social norms, etc.
  - **Architectural requirements**: distribution constraints, architectural styles, etc.
  - **Development requirements**: development cost, programming languages, tools, etc.

Non-functional Requirements

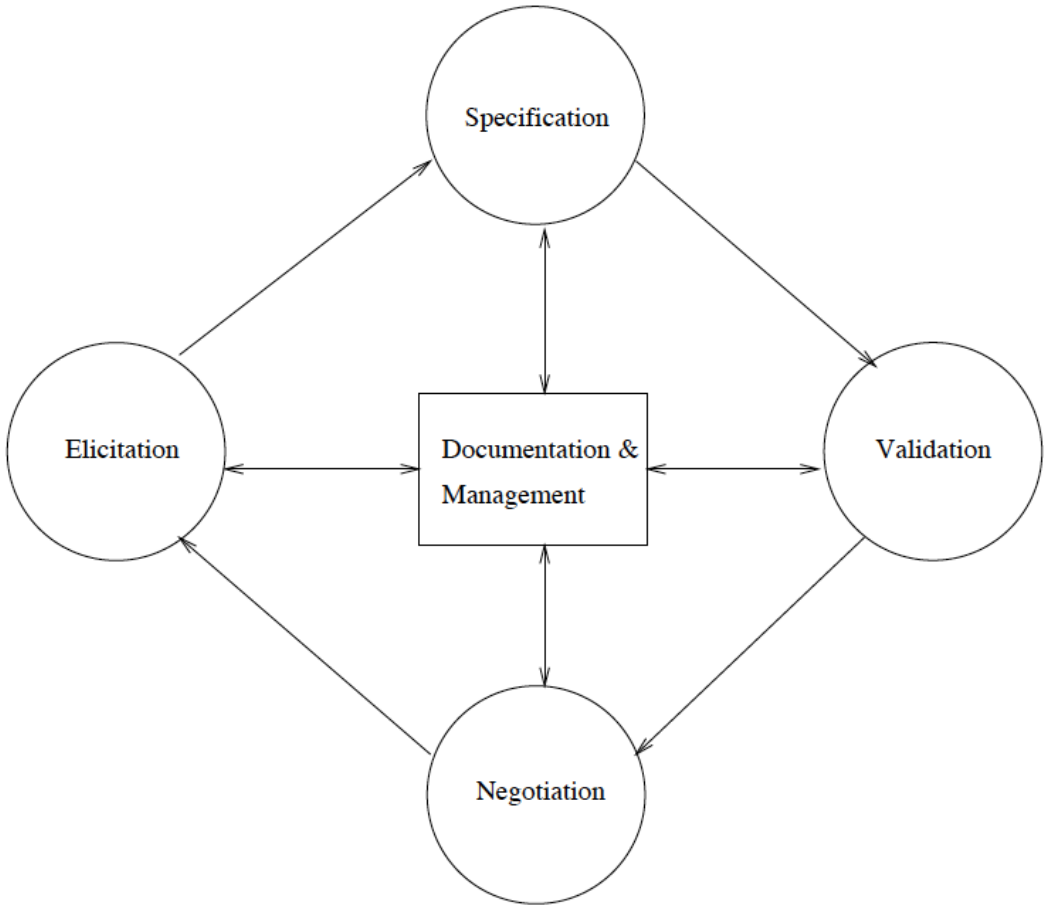


Examples of Non-functional Requirements

- **Safety requirements** are quality requirements that rule out software effects that might result in accidents, degradations or losses in the environment.
- **Security requirements** are quality requirements that prescribe the protection of system assets against undesirable environment behaviors.
- **Reliability requirements** constrain the software to operate as expected over long periods of time. Its services must be provided in a correct and robust way in spite of exceptional circumstances.
- **Interface requirements** are quality requirements that constrain the phenomena shared by the software-to-be and the environment.
- **Accuracy requirements** are quality requirements that constrain the state of the information processed by the software to reflect the state of the corresponding physical information in the environment accurately.

Requirements Engineering Process

- **Requirements Elicitation and Analysis**: collecting the requirements and understanding the problem.
- **Requirements Specification**: describing the product to be delivered.
- Requirements Validation and Verification: We have to ascertain that the correct requirements are stated (validation) and that these requirements are stated correctly (verification).
- Requirements Negotiation: resolve conflicts that stakeholders may have.



Requirements Elicitation: Techniques

- Identifying **stakeholders**
- Artifact-driven elicitation (domain understanding)
  - Background study
  - Data collection
  - **Questionnaires**
  - Prototypes and mock-ups
- Stakeholder-driven elicitation
  - **Interviews**
  - Observation and ethnographic studies
  - Group sessions

Stakeholder Analysis

- Stakeholders: groups or individuals affected by the system-to-be, who may influence its elaboration and its acceptance
  - Decision makers, managers, domain experts, users, clients, subcontractors, analysts, developers, ...
- Selection criteria:
  - Relevant position in the organization
  - Effective role in making decisions about the system-to-be
  - Level of domain expertise
  - Exposure to perceived problems
  - Influence in system acceptance
  - Personal objectives and conflicts of interest

Elicitation Techniques

- **Questionnaire:** a useful complement to interviews**.**
- **Interviewing** stakeholders (e.g., customers) by the requirements engineering team (i.e., requirements analyst).
  - Process: select stakeholders, schedule a meeting, write a report, submit the report to the interviewee for approval.
  - Structured interview and unstructured interview.
  - Can be difficult to elicit domain knowledge through interviews.
  - Should be used in conjunction with other requirements elicitation techniques.
- Observe customer (i.e., ethnography).
  - Groups of people are studied in their natural settings.
  - User requirements can be studied by participating in their daily work for a period of time.
  - Often used together with interviews.

Surveys and Questionnaires

- Submit a list of questions to selected stakeholders, each with a list of possible answers (+ brief context if needed)
  - Multiple choice question: one answer to be selected from answer list
  - Weighting question: qualitatively (e.g., “high”, “low”) or quantitatively (percentages).

- Effective method for collecting requirements when
  - collecting a small amount of data from a large number of people
  - people geographically distributed

- Helpful for preparing better focused interviews
- The acquired information may be biased. Why?

Guidelines for questionnaire design/validation

- Select a representative, statistically significant sample of people; provide motivation for responding
- Check coverage of questions, of possible answers
- Make sure questions, answers, formulations are unbiased & unambiguous
- Add implicitly redundant questions to detect inconsistent answers
- Have your questionnaire checked by a third party

Interviews

- Primary technique for knowledge elicitation
  1. Select stakeholder specifically for info to be acquired
     (domain expert, manager, salesperson, end-user, consultant, ...)
  2. Organize meeting with interviewee, ask questions, record answers
  3. Write report from interview transcripts
  4. Submit report to interviewee for validation & refinement
- Single interview may involve multiple stakeholders

  ☺ saves times

  ☹ weaker contact; individuals less involved, speak less freely
- The easiest yet most powerful techniques available for gathering requirements.

Types of interview

- Structured interview: predetermined set of questions
  - specific to purpose of interview
  - some open-ended, others with pre-determined answer set

  => more focused discussion, no rambling among topics

- Unstructured interview: no predetermined set of questions
  - free discussion about system-as-is, perceived problems, proposed solutions

  => exploration of possibly overlooked issues

=> Effective interviews should mix both modes ...

  - start with structured parts
  - shift to unstructured parts as felt necessary

Interviews: strengths & difficulties

☺ May reveal info not acquired through other techniques

- how things are running *really*, personal complaints, suggestions for improvement, ...

☺ On-the-fly acquisition of info appearing relevant

- new questions triggered from previous answers

☹ Acquired info might be subjective (hard to assess)

☹ Potential inconsistencies between different interviewees

☹ Effectiveness critically relies on interviewer's attitude, appropriateness of questions

=> *Interviewing guidelines*

Guidelines for effective interviews

- Preparing for an interview
  - Background study first
  - Know your stakeholders (e.g., name, role, primary responsibility)
  - pre-design a sequence of questions for the interviewee
- During the interview
  - Identify problems (e.g., Why is this a problem?)
  - Identify non-functional requirements (e.g., What are your expectations for system performance?)
  - Identify users (e.g., What computer skills do the users have?)
  - 'Must have' requirements, top priorities, scenarios to test, etc.
- After the interview
  - Edit & structure interview transcripts while still fresh in mind.

Guidelines for effective interviews

- Identify the right interviewee sample for full coverage of issues
  - different responsibilities, expertise, tasks, exposure to problems
- Come prepared, to focus on right issue at right time
  - background study first
  - pre-design a sequence of questions for this interviewee
- Focus on the interviewee's work & concerns
- Keep control over the interview
- Make the interviewee feel comfortable
  - *Start:* break ice, provide motivation, ask easy questions
  - Consider the person too, not only the role
  - Do always appear as a trustworthy partner

Guidelines for effective interviews

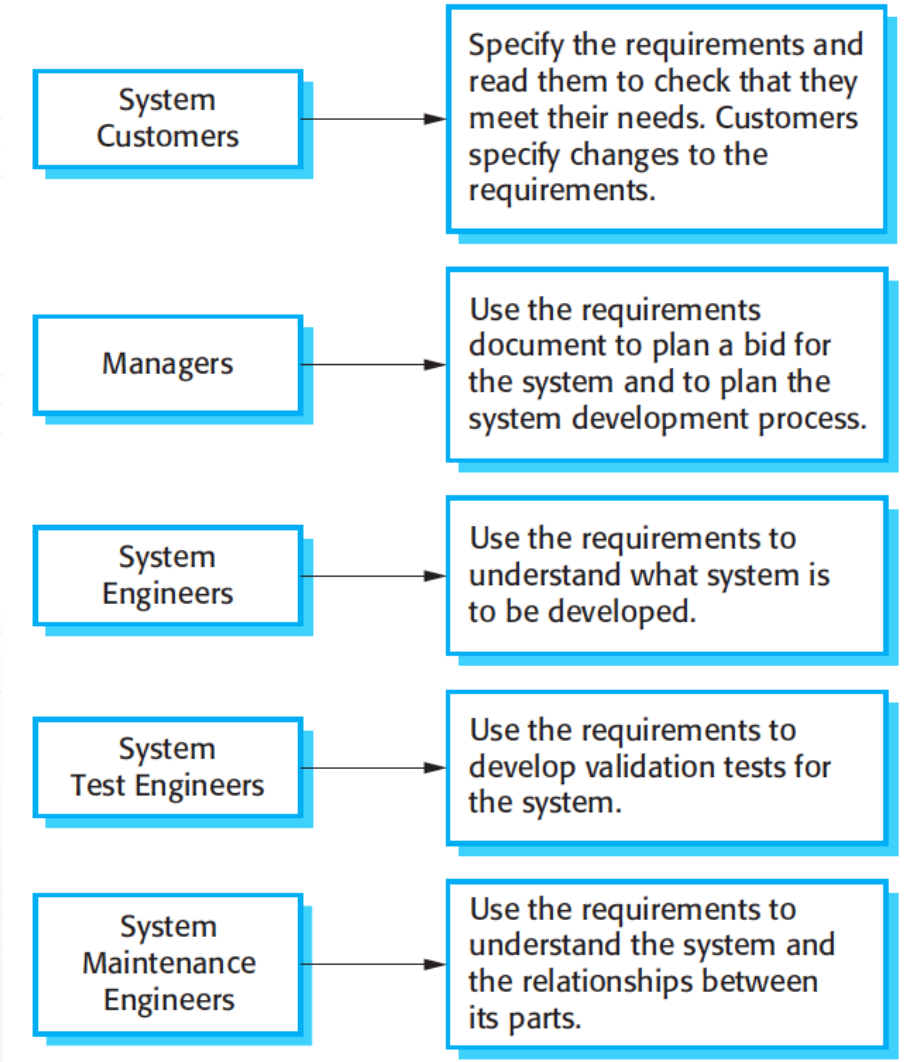
- Be focused, keep open-ended questions for the end
- Be open-minded, flexible in case of unexpected answers
- Ask *why*-questions without being offending
- Avoid certain types of questions ...
  - opinionated or biased
  - affirmative
  - obvious or impossible answer for this interviewee
- Edit & structure interview transcripts while still fresh in mind
  - including personal reactions, attitudes, etc
- Keep interviewee in the loop
  - co-review interview transcript for validation & refinement

Prioritizing Requirements

- **Must haves**: these are the top priority requirements, the ones that definitely have to be realized in order to make the system acceptable to the customer.
- **Should haves**: these requirements are not strictly mandatory, but they are highly desirable.
- **Could haves**: if time allows, these requirements will be realized as well. In practice, they usually won't.
- **Won't haves**: these requirements will not be realized in the present version. They are recorded though. They will be considered again for a next version of the system.

Requirements Specification

- Has a diverse set of users, who use the specification for different purposes.
- The level of detail depends on the type of system being developed.



Requirements Specification: Criteria

- **Correct**: The specification can be validated against other documents and by the end-user.
- **Unambiguous**: We must be able to uniquely interpret requirements.
- **Complete:** All significant matters relating to functionality, performance, constraints, and the like, should be documented.
- **Consistent**: Different parts of it should not be in conflict with each other.

Requirements Specification: Criteria

- **Ranked for importance or stability**
- **Verifiable:** There must be a finite process to determine whether or not the requirements have been met.
- **Modifiable:** The specification must be easy to change**.**
- **Traceable:** The origin and rationale of each and every requirement must be traceable. A clear and consistent numbering scheme makes it possible that other documents can uniquely refer to parts of the requirements specification.

More on User Requirements

- Invent a **standard format** and ensure all requirements adhere to that format.
- Use language **consistently**.
- Use **text highlighting** to pick out key parts of the requirement.
- Avoid the use of computer jargon (e.g., special words).