CSc 131
Computer Software Engineering

Chapter 7
SWE Requirements

Herbert G. Mayer, CSU CSc

Status 10/24/2019

Syllabus

- Objectives
- Types of Requirements
- Domain Requirements
- Requirement Guidelines
- Summary
- References

Objectives

Objectives

- To introduce the concepts of user- and system requirements
- To describe functional and non-functional requirements
- To explain how software (SW) requirements may be organised in a requirements document
- To describe requirements engineering, related to design activities and product processes

Topics Covered

- **Functional and non-functional requirements**
- **User requirements**
- **System requirements**
- **Interface specification**
- **Software requirements document**
- **Requirements engineering activities and formal processes**
- Use common sense to decide on a suitable level of requirements and specs! How many engineering resources will you spend on documentation + plans?

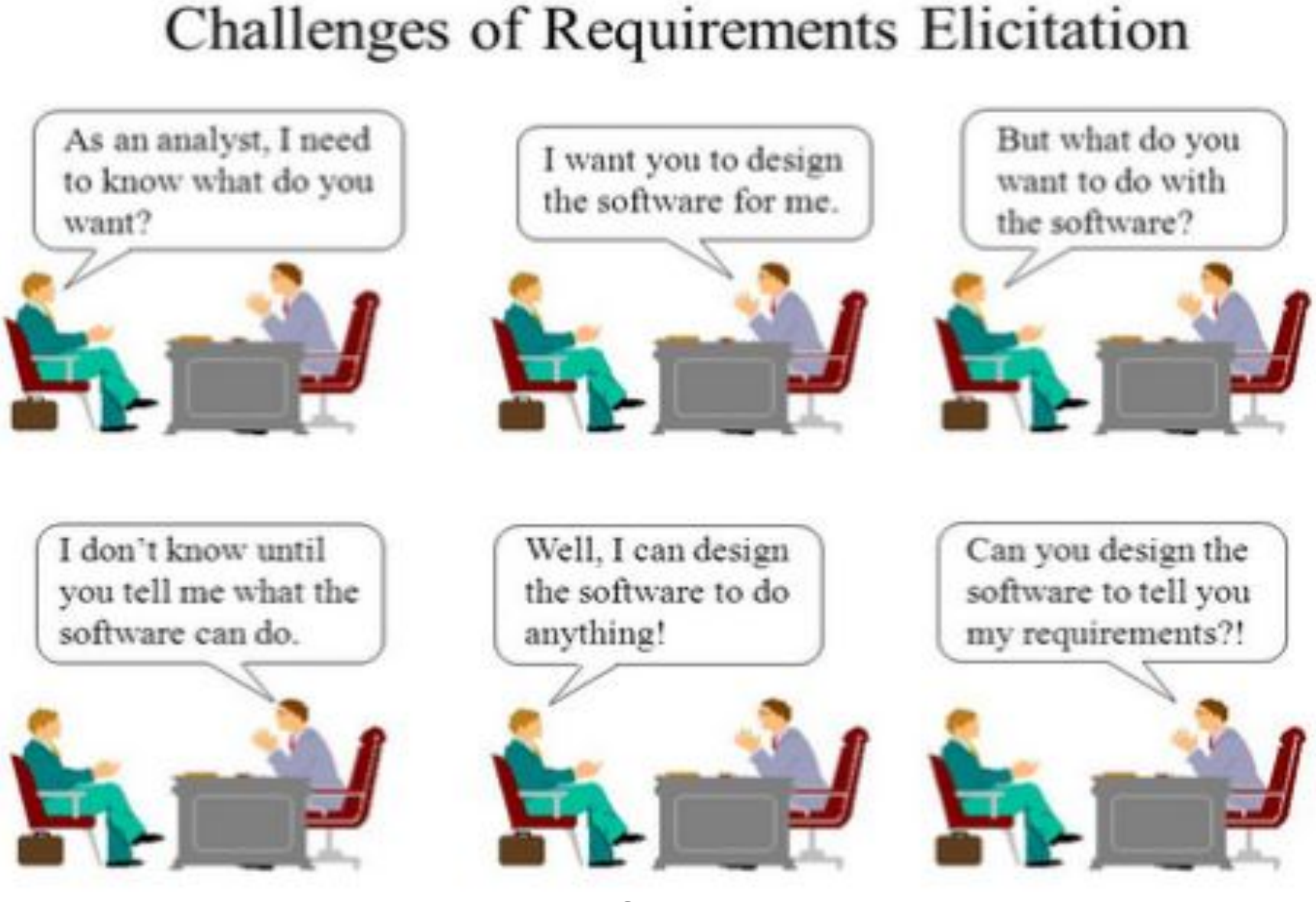
Requirements Engineering

- ***Requirements Engineering***: Addresses SWE process of *establishing* those services that customer requires from SW system, and constraints under which it operates
- Requirements themselves are stated descriptions of the system services and related constraints, generated during the requirements engineering process
  - Such descriptions are formally agreed about!
  - And documented
  - Ideal, if not changed during development; happens rarely that they remain static!

Requirements Engineering

- Specifying requirements may range from a high-level abstract statement of a service or system constraint, down to a detailed functional, even mathematical specification
- This effort is necessary and worthwhile, as requirements may serve a dual function:
  - May be the basis for a bid for a contract – therefore should be be open to interpretation
  - May be the basis for the contract itself - therefore must be defined in detail
  - Both of these statements (documents) may be referred to as: requirements

Requirements Elicitation



Requirements Abstraction

"If a company wishes to let a contract for a large SW development project, it must define its needs in a sufficiently abstract way that a solution is not pre-defined.

The requirements must be written so that several contractors can bid for the contract, offering, perhaps, different ways of meeting the client organisation's needs.

Once a contract has been awarded, the contractor must write a system definition for the client in more detail so that the client understands and can validate what the SW will do.

Both of these documents may be called the *requirements document* for the system."

Types of Requirements

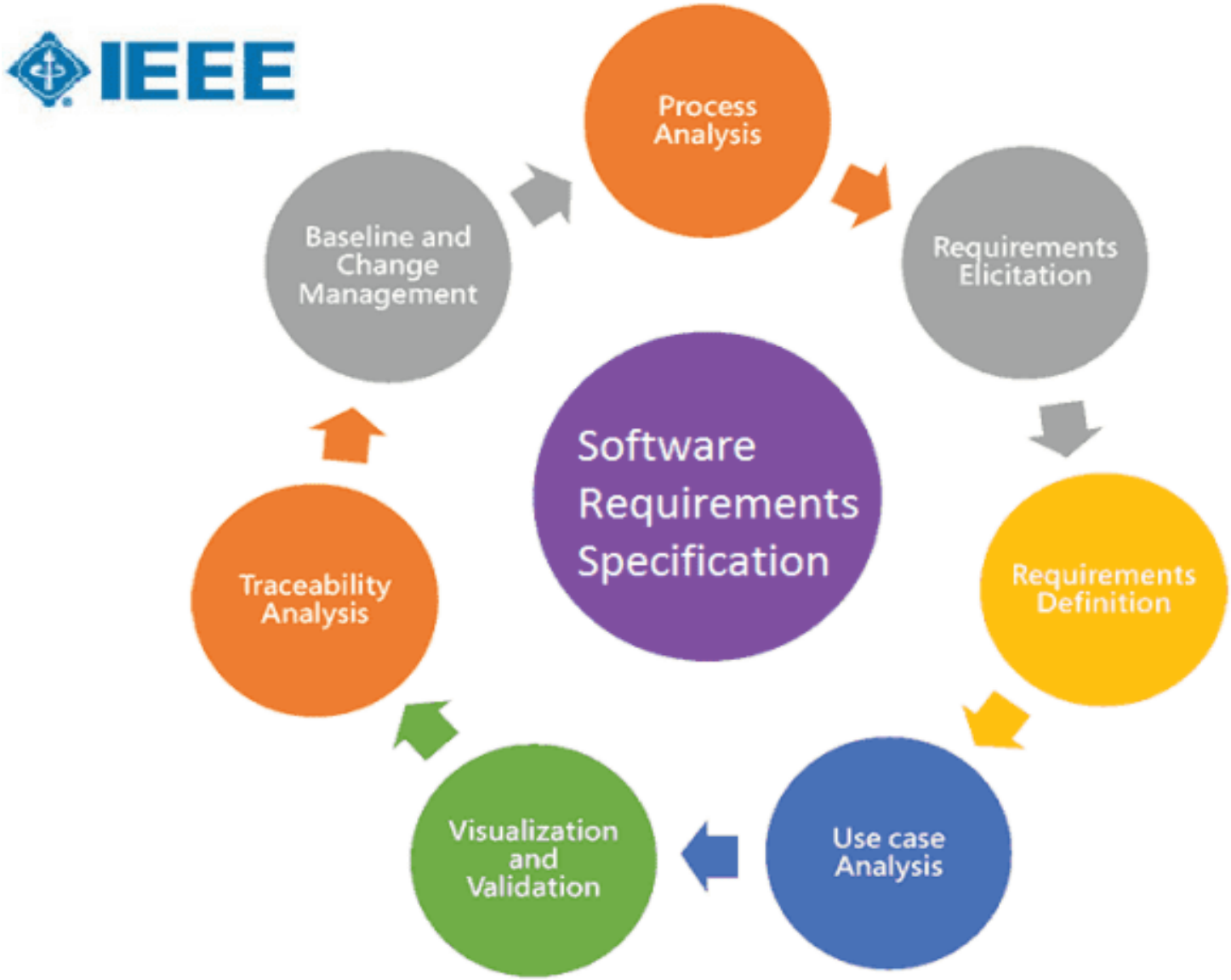
User requirements

- Statements in natural language plus diagrams describing the services that a system provides, and outlining its operational constraints
- Written for customer, by SWE organization

System requirements

- A structured document in some formal language describing system functions, services and operational constraints
- Defines what will be implemented, and what not
- May become part of contract between client and SWE contractor

Requirements Spec Per IEEE



Definition & Specification Sample

User requirement definition

1. The software must provide a means of representing and accessing external files created by other tools.

System requirements specification

1.1 The user should be provided with facilities to define the type of external files.
1.2 Each external file type may have an associated tool which may be applied to the file.
1.3 Each external file type may be represented as a specific icon on the user's display.
1.4 Facilities should be provided for the icon representing an external file type to be defined by the user.
1.5 When a user selects an icon representing an external file, the effect of that selection is to apply the tool associated with the type of the external file to the file represented by the selected icon.

Functional & Non-Functional Req's

Functional requirements

- Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in specific situations

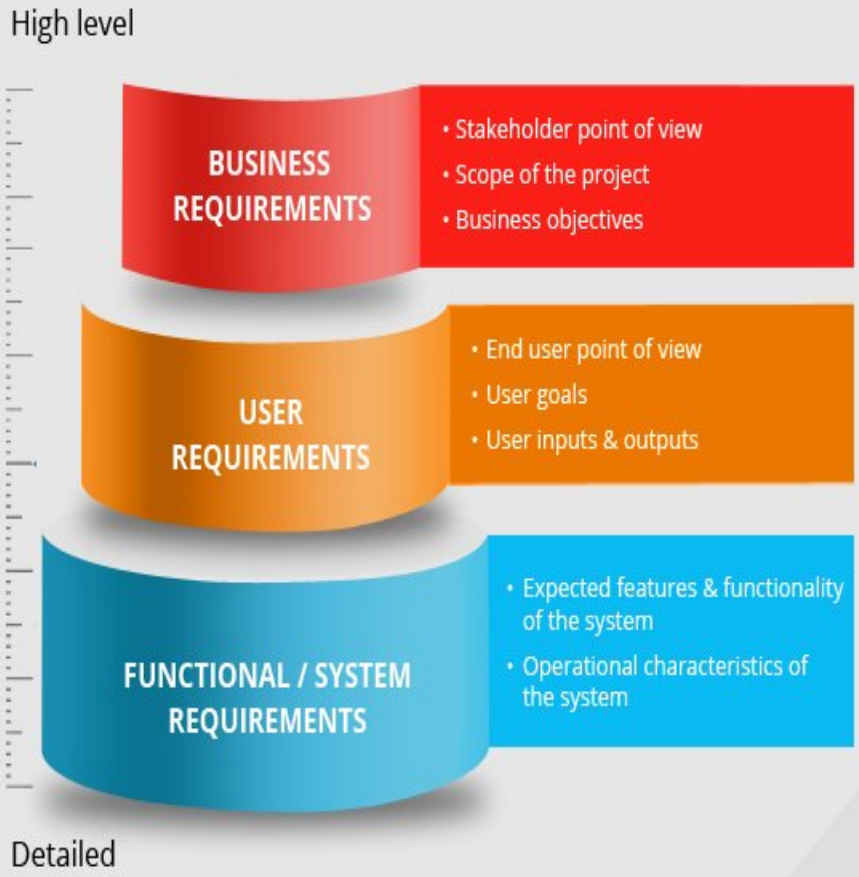
Non-functional requirements

- Constraints on services or functions offered by the system; e.g. timing constraints, constraints on the development process, referenced standards, etc.

Domain requirements

- Includes basic functions a system must exhibit in a domain
- E.g. Sac State's student records: The ability to access students by grade (faculty, name, level, …) is a domain requirement

Functional & Other Requirements



Functional Requirements

- **Describe functions or system services**
  - **Depend on type of SW, and depends on expected users and on type of system that uses the SW**
- **Functional *user requirements* may be *high-level* statements of what the system should do, as experienced by user**
  - **Which user command generates what response**
- **Functional *system requirements* describe the system services in detail, with limits and constraints**
  - **E.g. what are system limitations**
  - **Enumerate what the system cannot do**

An Example: LIBSYS

The LIBSYS System

- **Multiple companies that offer SW products, operating under this joint name: LIBSYS**
- **One company for example from India:**

  http://www.libsys.co.in

- **"LIBSYS Ltd. is the *leading provider* in Library Management Systems across India and offers solutions such as Library automation system, RFID, Digital Resource Management System etc. We plan, deploy, sustain and enhance your Library with *continuous innovations* in library automation. Our library software brings you a high level of certainty. The solutions offered by us are delivered on-time, are within your budget, and come with high quality, better efficiency, and responsiveness."**

LIBSYS US Logo

The LIBSYS System

- **A library system that provides a single interface to a number of databases of *articles* in different libraries**
- **Users can search for, download and print these *articles* for personal study**
- **Using one family of tools, but accessing varying databases or libraries**
  - **I.e. standardized database access, though actual databases themselves are (potentially) all different**
- **Numerous SW service companies exist under this same name LIBSYS, operating across the world**

Sample Functional Requirements (FR)

Some select, detailed FR samples:

1. **The user shall be able to search either all of the initial set of databases or select a subset from it**
2. **The system shall provide *appropriate viewers* for the user to read documents in the document store**
3. **Every order shall be allocated a unique identifier (ORDER_ID) which the viewer/user shall be able to copy to the account's permanent storage area**

Requirements Imprecision

- **Problem when requirements are not precisely stated**
- **Ambiguous requirements may be and will be interpreted in different ways by developers and users**
- **Consider the term '*appropriate viewers*'**
  - **User intention - special purpose viewer for each different document type**
  - **Developer interpretation - Provide a "text viewer" that shows the contents of the document, readable by people**

Requirement Imprecision

- In principle, the Requirements Specification should be complete and consistent
- Complete:
  - Include full descriptions of all facilities required
  - No omissions, but enumerate explicitly what is not provided
- Consistent:
  - There should be no conflicts or contradictions in the descriptions of a system's capabilities
- In practice, perfection is quite hard to impossible
- Impossible to always produce a complete and consistent requirements document

Non-Functional Requirements

- Non-functional requirements define system properties and constraints e.g. visual appeal, response speed, and storage requirements
- Sample constraints are I/O device capability, system representations
- Process requirements may also be specified mandating a particular CASE system, programming language or development method
- Non-functional requirements may at times be more critical than functional requirements; and if not met, system may be useless

Non-Functional Classifications

Product requirements

- Specify that/how the delivered product must behave in a particular way; e.g. execution speed, reliability, etc.

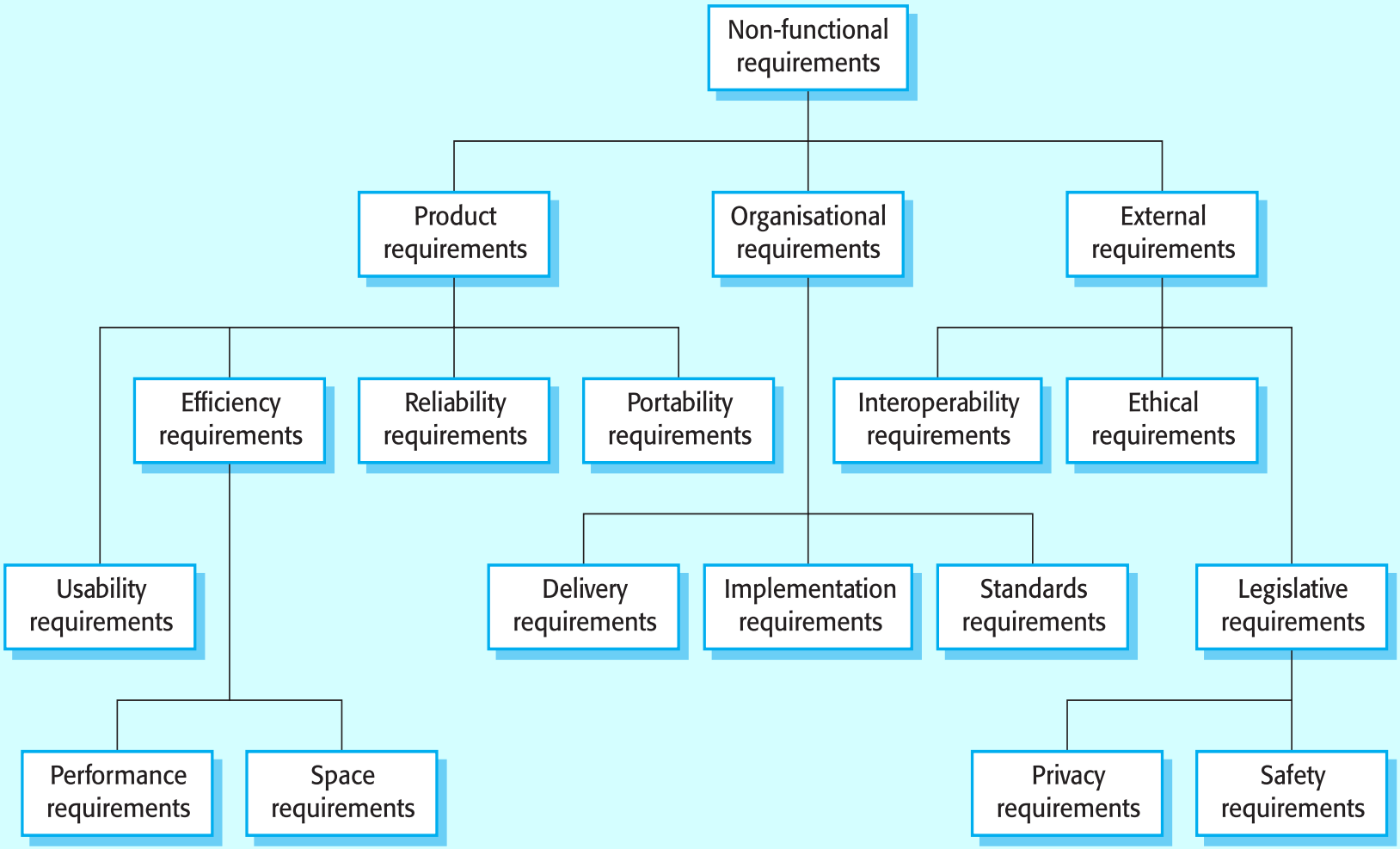
Organisational requirements

- Requirements which are a consequence of organisational policies and procedures
- E.g. process standards used, implementation requirements such as computers, P. languages, etc.

External requirements

- Requirements which arise from factors external to the system and its development process
- E.g. interoperability requirements, legislative requirements

Non-Functional Requirement Types



Non-Functional Requirement Samples

- **Product requirement** (see ref [3])
  - **8.1 The user interface for LIBSYS shall be implemented as simple HTML without frames or Java applets**
- **Organisational requirement**
  - **9.3.2 The system development process and deliverable documents shall conform to the process and deliverables defined in . . .**
- **External requirement**
  - **7.6.5 The system shall not disclose any personal information about customers apart from their name and reference number to the operators of the system**

Requirement Measures

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

Requirement Interactions

- Conflicts between different non-functional requirements are common in complex systems
- Spacecraft system
  - To minimise weight, the number of separate chips in the system should be minimised
  - To minimise power consumption, lower power chips should be used
  - However, using low power chips may mean that more chips have to be used
  - Which one is the critical and thus driving requirement?

Domain

- https://en.wikipedia.org/wiki/Domain_name
- "A domain name is an identification string that defines a realm of administrative autonomy, authority or control within the Internet. Domain names are used in various networking contexts and for application-specific naming and addressing purposes."
- "Domain requirements are the requirements which are characteristic of a particular category or domain of projects. The basic functions that a system of a specific domain must necessarily exhibit come under this category."

Domain Requirements

- Derived from the application domain and describes system characteristics and features that reflect that domain
- Domain requirements can be new functional requirements, and constraints on existing requirements or define specific computations
- If domain requirements of final system are not satisfied, such a system may be not useful

Domain Requirement Problems

Comprehension

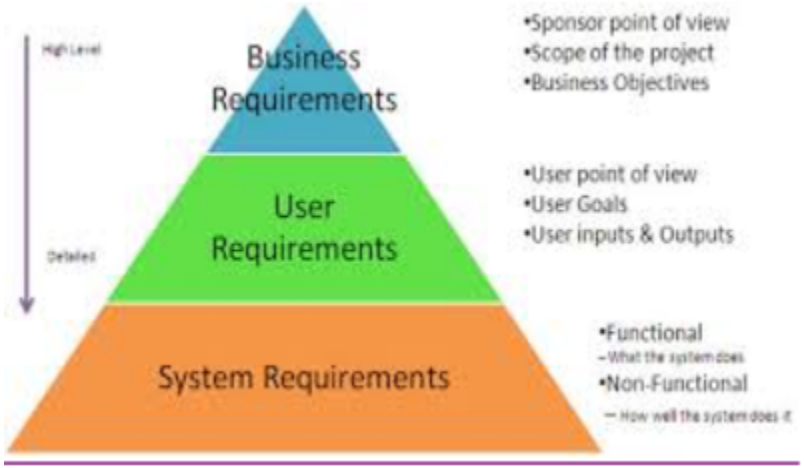
- Requirements are expressed in the language of the application domain
- This is often not understood by software engineers developing the system

Implicitness

- Domain specialists understand the area so well that they do not think of making the domain requirements explicit
- Problem with assumptions we are unaware of

User Requirements

- **Should describe functional and non-functional requirements in such a way that they are understandable by system users who lack detailed technical knowledge**
- **User requirements are defined using natural language, tables and diagrams as these can be understood by all users**



Problems with NL Natural Language

Lack of clarity

- Precision is difficult without *making hard to* read documents

Requirements confusion

- Functional and non-functional requirements tend to be mixed-up

Requirements amalgamation

- Several different requirements may be expressed together

Ambiguity

- Multiple meanings
- for same phrase

Requirement Guidelines

- Invent and communicate a standard format and use it for all requirements
- Use language in a consistent way
- Use shall for mandatory requirements, should for desirable requirements
- Use text highlighting to identify key parts of the requirement
- Avoid computer jargon

System Requirements

- More detailed specifications of system functions, services and constraints than user requirements
- They are intended to be basis for designing an overall system
- They may be incorporated into the system contract
- System requirements may be defined or illustrated using system models

Requirements & Design

- In principle, requirements should state what the system should do
- A design should describe how the system accomplishes this
- In practice, requirements and design are often inseparable
  - A system architecture may be designed to structure the requirements
  - The system may inter-operate with other systems that generate design requirements
  - The use of a specific design may be a domain requirement

Problems with NL Specification

Ambiguity

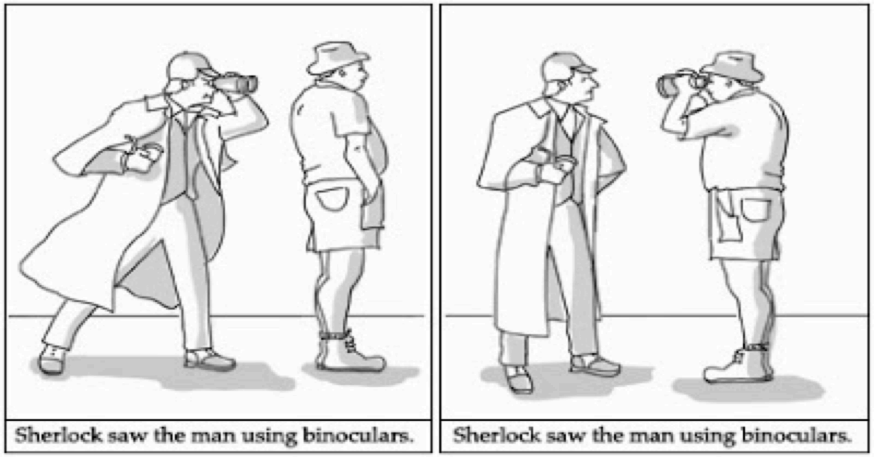
- The readers and writers of the requirement must interpret the same words in the same way
- Natural language (NL) inherently ambiguous so can be hard

Over-flexibility

- The same thing may be said in a number of different ways in the specification

Lack of modularisation

- NL structures are inadequate to structure system requirements



Alternatives to NL Specification

Notation	Description
Structured natural language	This approach depends on defining standard forms or templates to express the requirements specification.
Design description languages	This approach uses a language like a programming language but with more abstract features to specify the requirements by defining an operational model of the system. This approach is not now widely used although it can be useful for interface specifications.
Graphical notations	A graphical language, supplemented by text annotations is used to define the functional requirements for the system. An early example of such a graphical language was SADT. Now, use-case descriptions and sequence diagrams are commonly used .
Mathematical specifications	These are notations based on mathematical concepts such as finite-state machines or sets. These unambiguous specifications reduce the arguments between customer and contractor about system functionality. However, most customers don't understand formal specifications and are reluctant to accept it as a system contract.

Structured Language Specifications

- The freedom of the requirements writer is limited by a predefined template for requirements
- All requirements are written in a standard fashion
- The terminology used in the description may be reasonable contained (limited)
- Advantage: Most of the expressiveness of natural language is maintained but a degree of uniformity is imposed on the specification

Form-Based Specifications

- Definition of the function or entity
- Description of inputs and where they come from
- Description of outputs and where they go to
- Indication of other entities required
- Pre and post conditions (if appropriate)
- The side effects (if any) of the function

Form-Based Node Specification

Insulin Pump/Control Software/SRS/3.3.2	
Function	Compute insulin dose: Safe sugar level
Description	Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units.
Inputs	Current sugar reading (r2), the previous two readings (r0 and r1)
Source	Current sugar reading from sensor. Other readings from memory.
Outputs	CompDose – the dose in insulin to be delivered
Destination	Main control loop
Action:	CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result. If the result, is rounded to zero then CompDose is set to the minimum dose that can be delivered.
Requires	Two previous readings so that the rate of change of sugar level can be computed.
Pre-condition	The insulin reservoir contains at least the maximum allowed single dose of insulin..
Post-condition	r0 is replaced by r1 then r1 is replaced by r2
Side-effects	None

Tabular Specification

- Used to supplement natural language
- Particularly useful when you have to define a number of possible alternative courses of action
- Concise and easy to comprehend when there are multiple variations of one theme
- Tabular specification even a product at IBM Corp:

https://www.ibm.com/support/knowledgecenter/en/SSMLQ4_11.3.0/com.ibm.nex.optimd.common.doc/04acdefs/opcommon-r-table_specifications.html

Tabular Specification

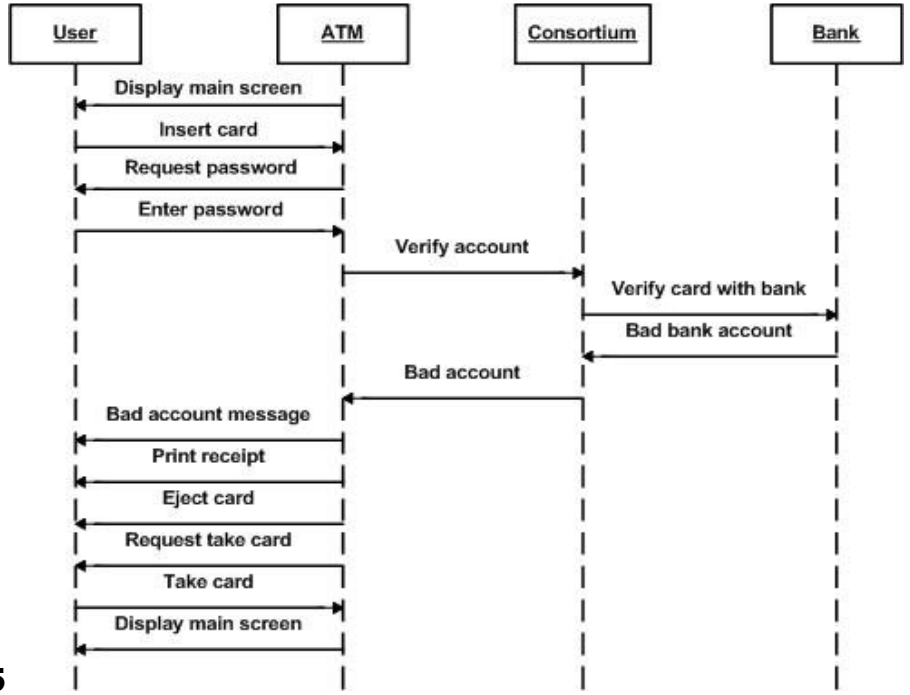
Condition	Action
Sugar level falling (r2 < r1)	CompDose = 0
Sugar level stable (r2 = r1)	CompDose = 0
Sugar level increasing and rate of increase decreasing ((r2-r1)<(r1-r0))	CompDose = 0
Sugar level increasing and rate of increase stable or increasing. ((r2-r1) ≥ (r1-r0))	CompDose = round ((r2-r1)/4) If rounded result = 0 then CompDose = MinimumDose

Graphical Models

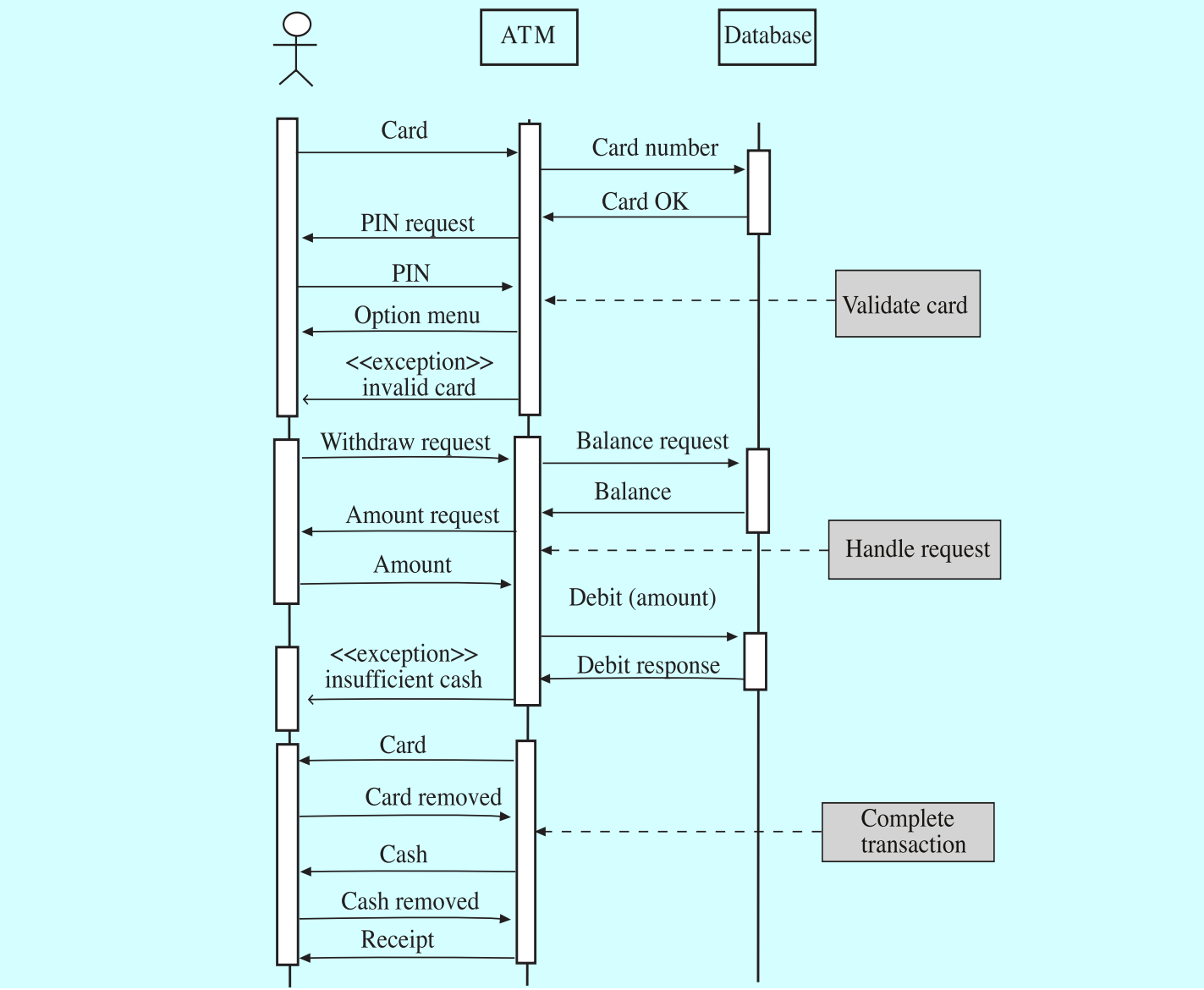
- Graphical models useful to show how state changes
- Or to describe a sequence of actions
- Picture often way more concise
- Picture is worth 1000 words, yet it takes words to say!

Sequence Diagrams

- These show the sequence of events that take place during some user interaction with a system
- You read them from top to bottom to see the order of the actions that take place
- Cash withdrawal from an ATM
  - Validate card
  - Handle request
  - Complete transaction



Detailed Sequence at ATM Withdrawal



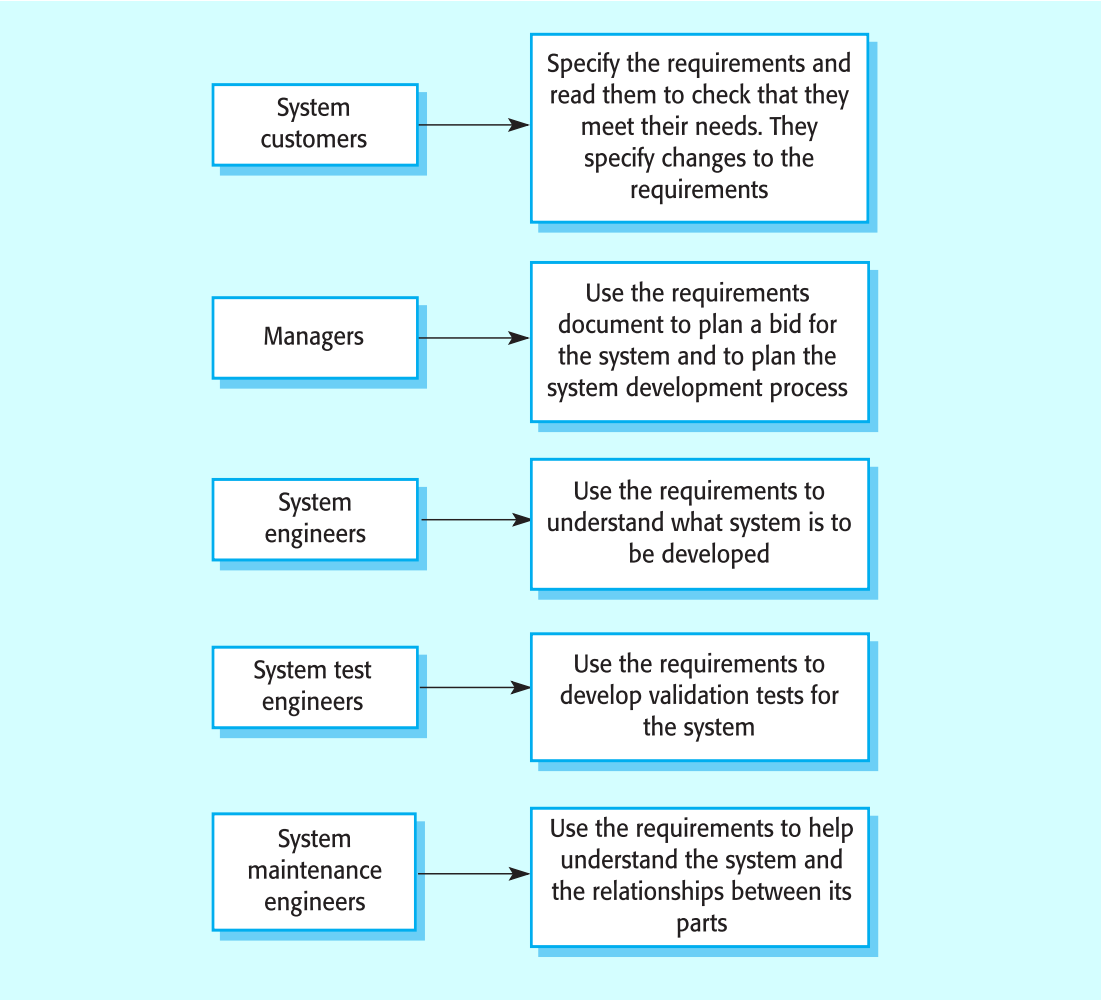
Interface Specification

- Most systems cooperate with other systems
- Thus operating interfaces must be specified as part of system requirements
- Typically, one of three types of interface may be defined:
  - Procedural interfaces –what to do
  - Data structures that are exchanged –which data to send/receive
  - Data representations –how to store data
- Formal notations are an effective technique for interface specification
- Could be by text, picture, other

Requirements Document

- A requirements document (RD) is the official statement of what is required to be crafted by system developers
- RD should include definition of user requirements and a specification of system requirements
- RD is NOT a design document
- But forms the logical basis for a to be crafted design document (DD)
- Where possible, RD should articulate what WHAT the system should do rather than HOW to do it

Consumer of Requirements Document



IEEE Requirements Standard

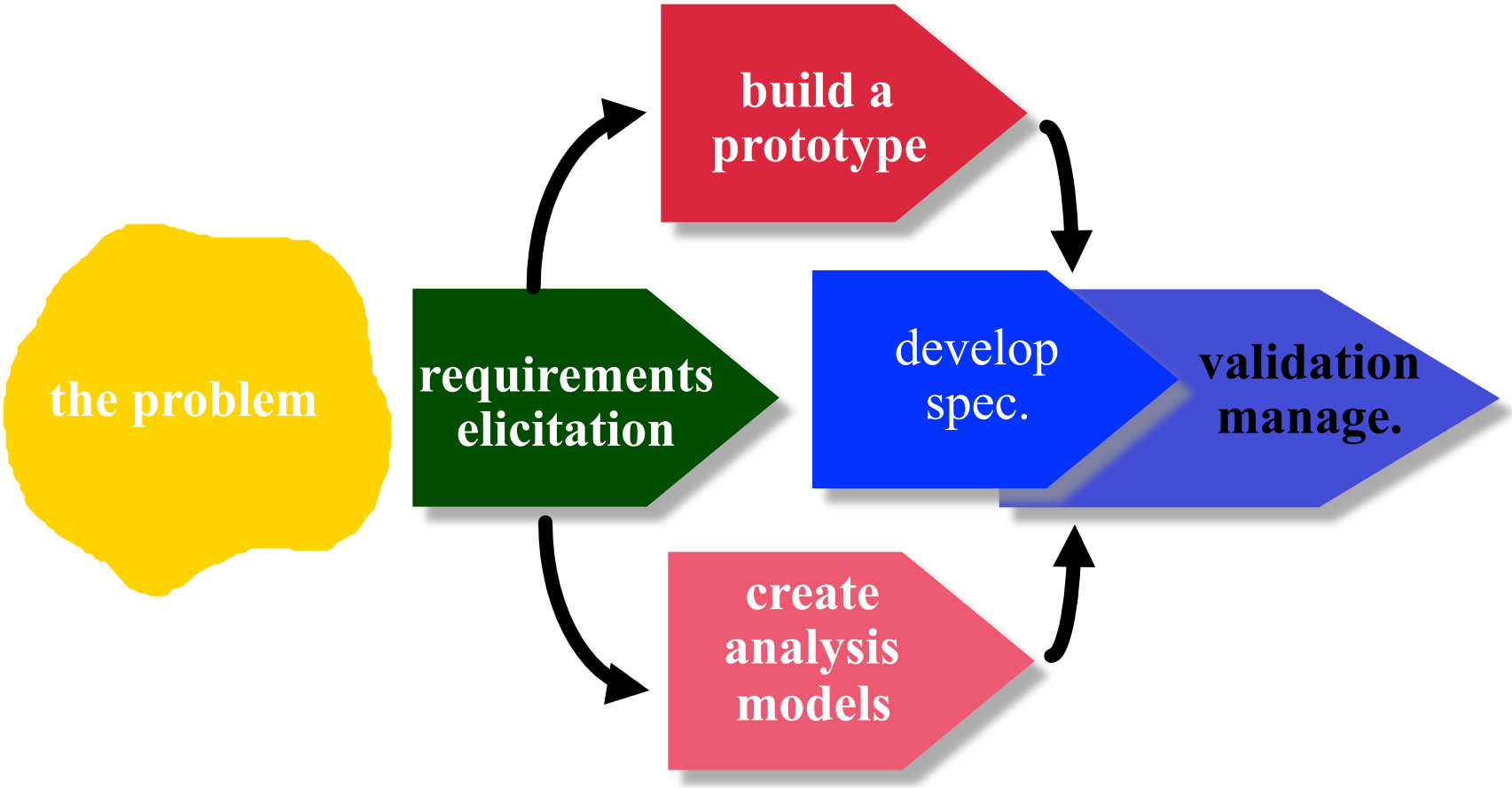
An IEEE standard defines a generic structure for a requirements document that must be instantiated for each specific system

- Introduction
- General description
- Specific requirements . . . The detail
- Appendices
- Index

Requirement Document Structure

- Preface
- Introduction
- Glossary
- User requirements definition
- System architecture
- System requirements specification
- System models
- System evolution
- Appendices
- Index

Requirements Analysis Process



Requirements Engineering

- **Inception** —ask relevant questions that establish
  - basic understanding of problem.
  - Interview the people who want a solution.
  - Identify the solution type that is desired, and
  - Initiate communication between the customer and developer
- **Elicitation** —elicit requirements from all stakeholders
- **Elaboration** —create an analysis model that identifies data, function and behavioral requirements
- **Negotiation** —agree on a deliverable system that is realistic for developers and customers

Requirements Engineering

Specification —can be any one (or more) of:

- **A written document**
- **A set of models**
- **A formal mathematical expression**
- **A collection of user scenarios (use-cases)**
- **A functioning prototype**

Requirements Engineering

Validation —a review mechanism that looks for

- errors in content or interpretation
- areas where clarification may be required
- missing information
- inconsistencies (a major problem when large products or systems are engineered)
- conflicting or unrealistic requirements.

Requirements management

Inception

Identify stakeholders

- "who else do you think I must talk with?"

Recognize multiple points of view

Work toward collaboration

The first questions:

- Who is behind the request for this work?
- Who will use the solution?
- What will be the benefit of a successful solution?
- Is there another source for the needed solution?
- What damage is done in case of lack of success?

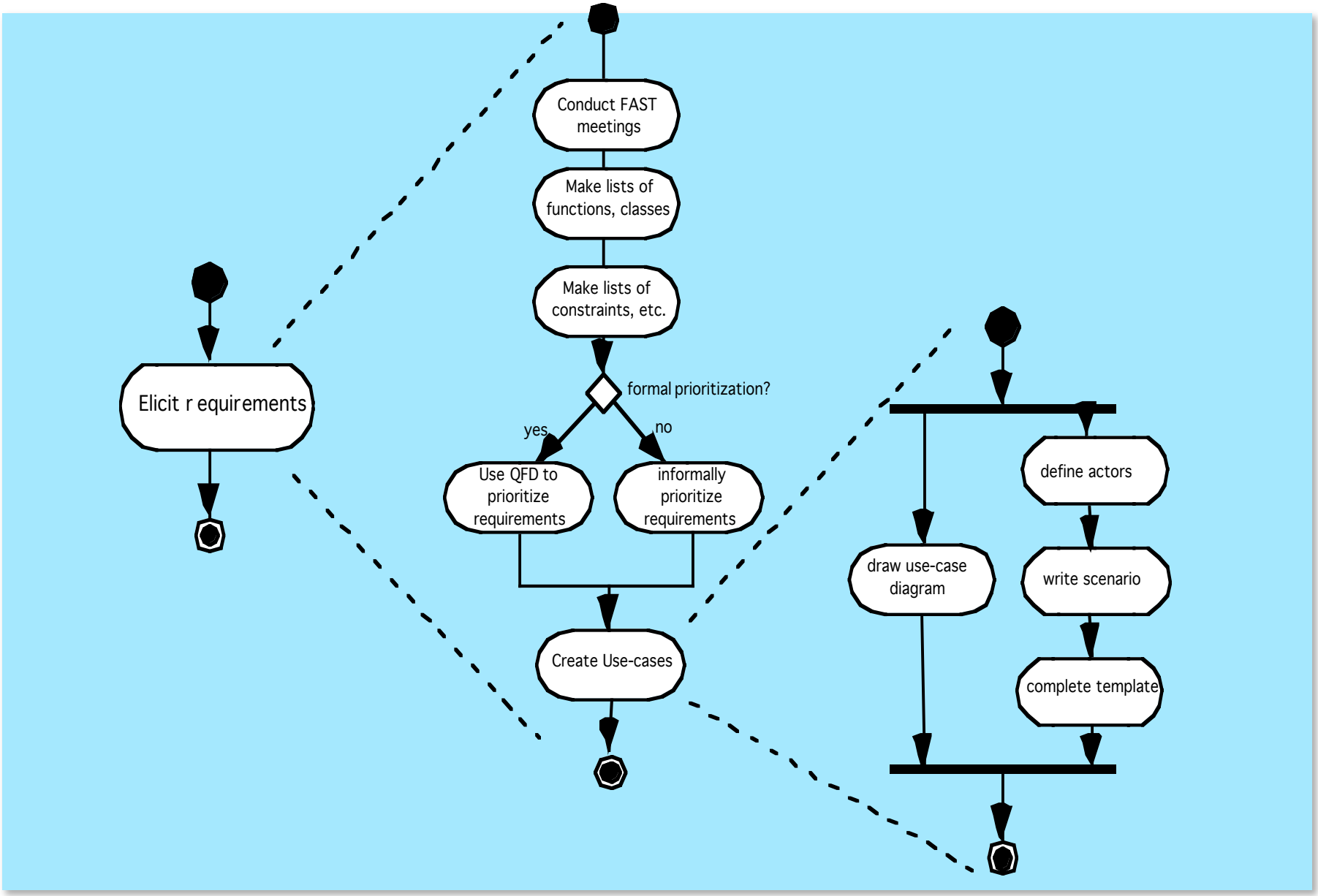
Eliciting Requirements

- Meetings are conducted and attended by both software engineers and customers: those who need product, and those who design, build product
- Rules for preparation and participation are established
- Agenda is suggested, published
- A "facilitator" (can be a customer, a developer, or an outsider) if defined, selected
- Controls meeting; facilitator's main function:
  - Focus on real agenda
  - Targeted use of time: no chit chat
  - Summary
  - Follow up?

Eliciting Requirements

- A definition mechanism is agreed upon, and used
- Can be:
  - work sheets
  - flip charts
  - wall stickers
  - electronic bulletin board
  - chat room or virtual forum
- Goal:
  - to identify the problem
  - propose elements of the solution
  - negotiate different approaches
  - specify a preliminary set of solution requirements

Eliciting Requirements



Quality Function Deployment

- **Function deployment** determines the "value" (as perceived by the customer) of each function required of the system
- **Information deployment** identifies data objects and events
- **Task deployment** examines the behavior of the system
- **Value analysis** determines the relative priority of requirements

Eliciting Work Products

- Statement of need and feasibility
- Bounded statement of scope for system or product
- List of customers, users, other stakeholders who participated in requirements elicitation
- Description of the system's technical environment
- List of requirements (
  - Organized by function) and
  - Domain constraints that apply to each
- Sample usage scenarios that provide insight into the use of the system or product under different operating conditions
- Prototypes developed to better define requirements

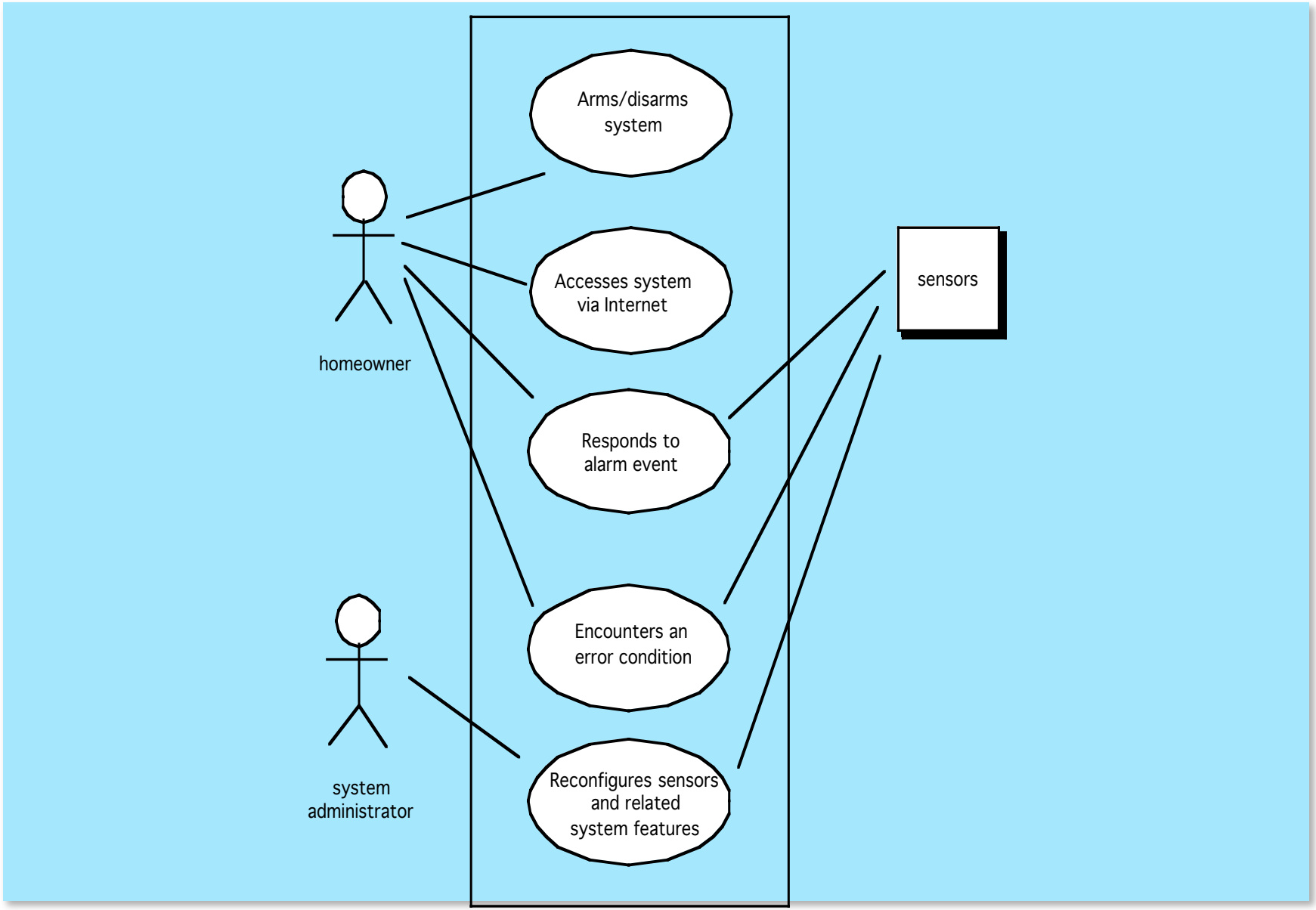
Use-Cases

- A collection of user scenarios that describe the thread of usage of a system
- Each scenario is described from the point-of-view of an “actor”—a person or device that interacts with the software in some way
- Each scenario answers the following questions:
  - Who is the primary actor, the secondary actor (s)?
  - What are the actor’s goals?
  - What preconditions should exist before the story begins?
  - What main tasks or functions are performed by the actor?
  - What extensions might be considered as the story is described?

Use-Cases

- **Each scenario answers the following questions, Cont'd:**
  - **What variations in the actor's interaction are possible?**
  - **What system information will the actor acquire, produce, or change?**
  - **Will the actor have to inform the system about changes in the external environment?**
  - **What information does the actor desire from the system?**
  - **Does the actor wish to be informed about unexpected changes?**

Use-Case Diagram



Building Analysis model

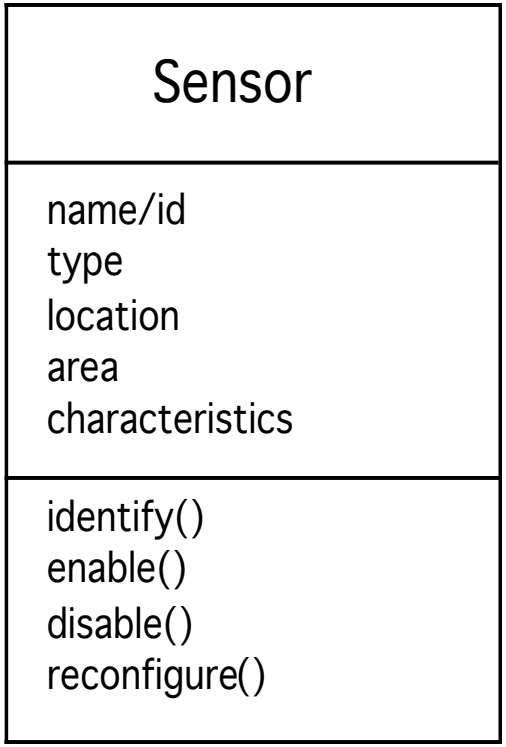
Elements of the analysis model

- Scenario-based elements
  - Functional—processing narratives for software functions
  - Use-case—descriptions of the interaction between an "actor" and the system
- Class-based elements
  - Implied by scenarios
- Behavioral elements
  - State diagram
- Flow-oriented elements
  - Data flow diagram

Class Diagram

From the *SafeHome* system

...



State Diagram

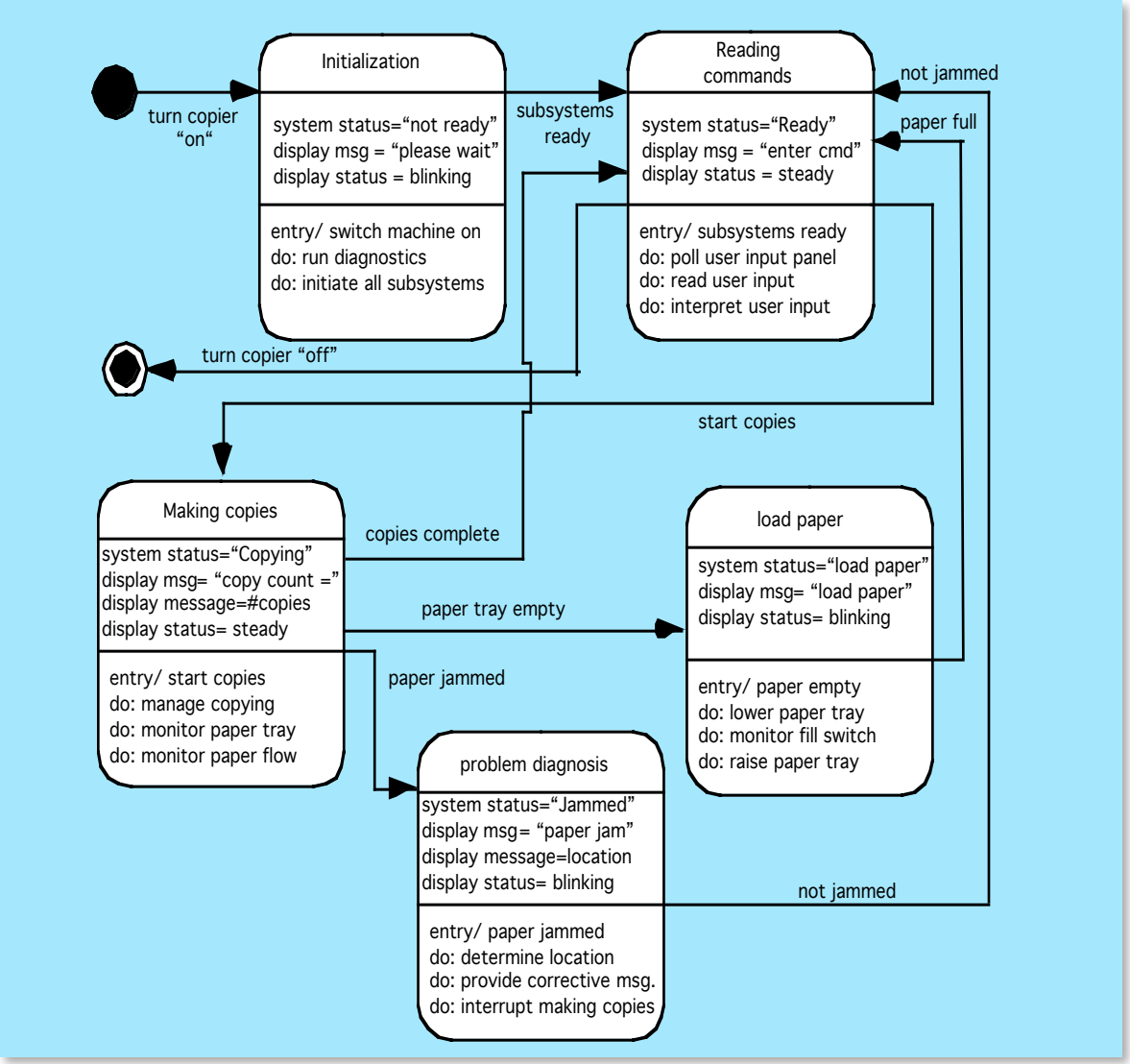


Figure 7.6 Preliminary UML state diagram for a photocopier

Analysis Patterns

Pattern name: A descriptor that captures the essence of the pattern.
Intent: Describes what the pattern accomplishes or represents
Motivation: Illustrates how pattern can be used to address problem.
Forces and context: A description of external issues affecting how pattern is used and external issues to be resolved when pattern is applied.
Solution: description how pattern is applied to solve problem, emphasis on structural and behavioral issues.
Consequences: Addresses what happens when pattern is applied and what trade-offs exist during its application.
Design: Discusses how the analysis pattern can be achieved through use of known design patterns.
Known uses: Examples of uses within actual systems.
Related patterns: Analysis patterns related to named pattern because (1) it is commonly used with the named pattern; (2) is structurally similar to named pattern; (3) is a variation of the named pattern.

Negotiating Requirements

- **Identify the key stakeholders**
  - **These are the people who will be involved in negotiations**
- **Determine each of the stakeholders "win conditions"**
  - **Win conditions are not always obvious**
  - **Are at times contradictory!**
- **Negotiate**
  - **Work toward a set of requirements that lead to "win-win"**
  - **Experienced SWE practices more than just complex SW development and productization!**

Validating Requirements

- Is each requirement consistent with the overall objective for the system/product?
- Have all requirements been specified at the proper level of abstraction? That is, do some requirements provide a level of technical detail that is inappropriate at this stage?
- Is the requirement really necessary or does it represent an add-on feature that is nice to have?
- Is each requirement bounded and unambiguous?
- Does each requirement have attribution? That is, is a source (generally, a specific individual) noted for each requirement?

Validating Requirements

- Do any requirements conflict with other requirements?
- Is each requirement achievable in the technical environment that will house the system or product?
- Is each requirement testable, once implemented?
- Does the requirements model properly reflect the information, function and behavior of the system to be built?
- Has the requirements model been "partitioned" in a way that exposes progressively more detailed information about the system?

Validating Requirements

- Have requirements patterns been used to simplify the requirements model?
- Have all patterns been properly validated?
- Are all patterns consistent with customer requirements?

Summary

- Document clearly all function requirements, prioritize, suggest areas for later upgrades
- Separate user, system, future requirements
- Various specification types: tabular, form-based, structured
- Start by eliciting requirements, particularly user requirements
- Include requirements in SW development contract

References

1. NL-based requirements: Umber A., Bajwa I.S., Asif Naeem M. (2011) NL-Based Automated Software Requirements Elicitation and Specification. In: Abraham A., Lloret Mauri J., Buford J.F., Suzuki J., Thampi S.M. (eds) Advances in Computing and Communications. ACC 2011. Communications in Computer and Information Science, vol 191. Springer, Berlin, Heidelberg
2. See: https://link.springer.com/chapter/10.1007/978-3-642-22714-1_4
3. For LIBSYS, see: https://www.libsysinc.com