U O W

Software Requirements, Specifications and Formal Methods

Dr. Shixun Huang

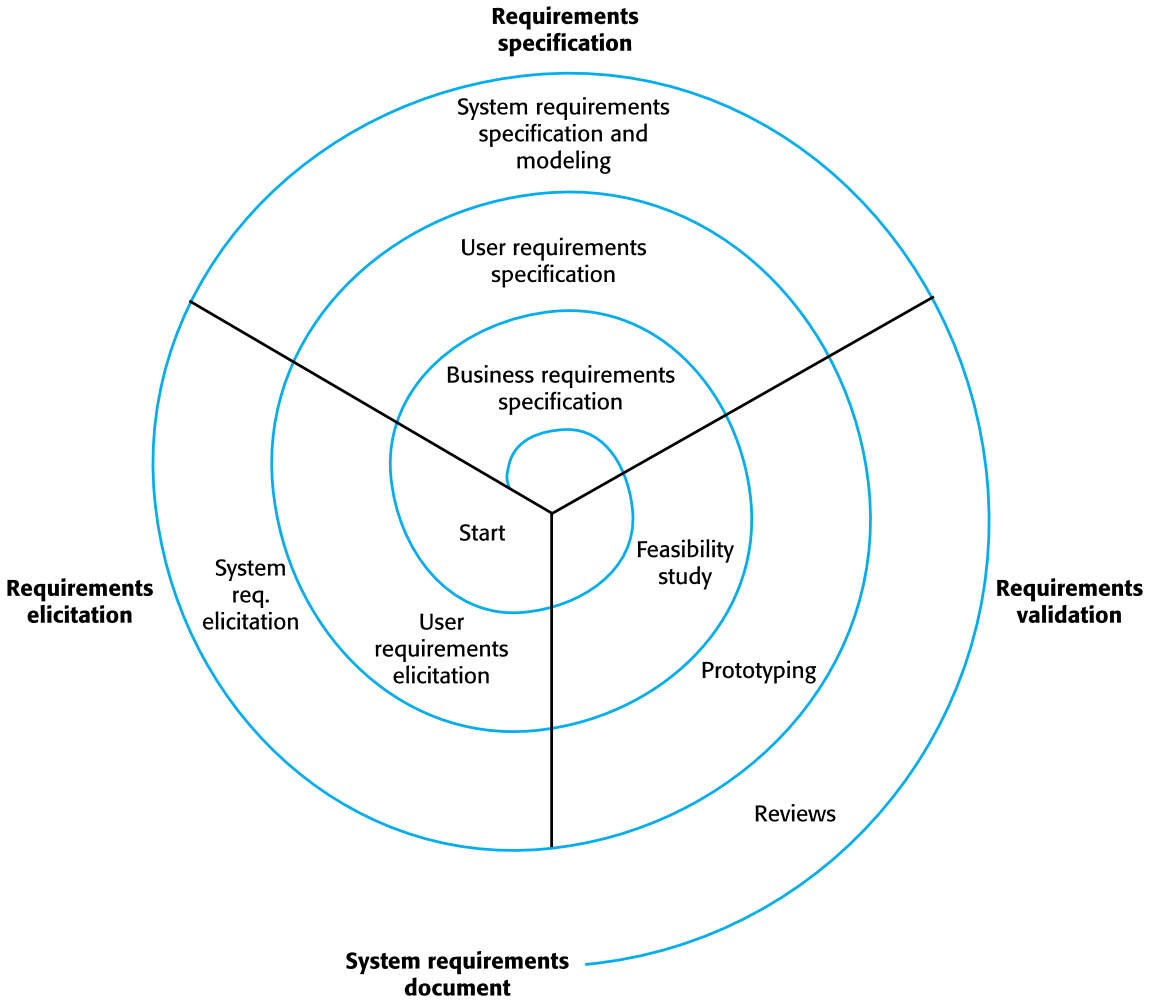
UNIVERSITY OF WOLLONGONG AUSTRALIA

Requirements Engineering Process

Requirements engineering processes

- The processes used for RE vary widely depending on the application domain, the people involved and the organisation developing the requirements.
- However, there are a number of generic activities common to all processes
  - Requirements elicitation/discovery
  - Requirements analysis and reconciliation
  - Requirements representation/modeling
  - Requirements verification and validation
  - Requirements management

A spiral view of the requirements engineering process



Requirements Elicitation/Discovery

- Sometimes called requirements elicitation or requirements discovery.
- Involves **technical staff** working with **customers** to find out about the **application domain**, the **services** that the system should provide and the system's operational **constraints**.
- May involve end-users, managers, engineers involved in maintenance, domain experts, trade unions, etc. These are called ***stakeholders***.

Requirements Analysis and Agreement

Requirements analysis and agreement involves techniques to deal with a number of problems with requirements in their “raw” form after they are collected from the customer. Problems with raw requirements include the following:

- ✧ They don’t always make sense.
- ✧ They often contradict one another (and not always obviously so).
- ✧ They may be inconsistent.
- ✧ They may be incomplete.
- ✧ They may be vague or just wrong.
- ✧ They may interact and be dependent on each other.

Requirements Representation

- Requirements representation (or modeling) involves converting the requirements processed raw requirements into some forms (usually natural language, mathematics, and visualizations).
- Proper representations facilitate communication of requirements and conversion into a system architecture and design.
- Various techniques are used for requirements representation including informal (e.g., natural language, sketches, and diagrams), formal (mathematically sound representations), and semiformal (convertible to a sound representation or can be made fully formal by the addition of a semantic framework).

Requirements Validation

- Requirements validation is the process of determining if the specification is a correct representation of the customers' needs.
- Validation answers the question "Am I building the right product?" Requirements validation involves various semiformal and formal methods, text-based tools, visualizations, inspections, and so on

Requirements Management

- Requirements management involves managing the realities of changing requirements over time.
- It also involves fostering *traceability* through appropriate aggregation and subordination of requirements and communicating changes in requirements to those who need to know.
- Managers also need to learn the skills to intelligently push back when *scope creep* ensues. Using tools to track changes and maintain traceability can significantly ease the burden of requirements management.

Requirements Engineer Roles

- ✧ Requirements engineer as software systems engineer
- ✧ Requirements engineer as subject matter expert (SME)
- ✧ Requirements engineer as architect
- ✧ Requirements engineer as business process expert

Role of the Customer

- Helping the requirements engineer understand what they need and want (elicitation and validation)
- Helping the requirements engineer understand what they don't want (elicitation and validation)
- Providing domain knowledge when necessary and possible
- Alerting the requirements engineer quickly and clearly when they discover that they or others have made mistakes
- Alerting the requirements engineer quickly when they determine that changes are necessary
- Controlling their urges to have "aha moments" that cause major scope creep
- Sticking to all agreements

Problems with Traditional Requirements Engineering

- ✧ Natural language problems (e.g., ambiguity, imprecision)
- ✧ Domain understanding
- ✧ Dealing with complexity (especially temporal behavior)
- ✧ Difficulties in *enveloping* system behavior
- ✧ Incompleteness (missing functionality)
- ✧ Over-completeness (gold-plating)
- ✧ Overextension (dangerous “all”)
- ✧ Inconsistency
- ✧ Incorrectness
- ✧ and more

Preparing for Requirements Elicitation

Product Mission Statement

- The first thing we need to do when undertaking the development of a new system, or redesign of an old one, is to obtain or develop a concise description of what it is supposed to do.
- Such a statement is often called a product mission statement (or system mission statement)
- The product mission statement allows us to weigh the importance of various features
- Writing mission statements can be a contentious business, and many people resent or fear doing so because there can be a tendency to get bogged down in minutiae.
- A product mission statement should be very short, descriptive, compelling, and never detailed.

Preparing for Requirements Elicitation

Encounter with Your Customers

- ✧ You need to understand the application domain, and the vocabulary used by your customers
- ✧ Customers don’t always know what they want
- ✧ Never make assumptions about what customers want
- ✧ Customers can change their mind
- ✧ They may have high expectations about what you know and what you will provide.

Identifying the System Boundaries

- A necessary first step in identifying stakeholders is to create a high-level systems model. This model will to identify the set of people and entities involved with the system and define the system boundary—the direct and indirect interactions with other entities (Laplante et al. 2016).
- A context diagram can be used

Stakeholders

Stakeholders represent the set of individuals who have some interest (a stake) in the success of the system in question. Typical stakeholders may include

- Customers
- Sponsors
- All responsible engineering and technical persons
- Regulators
- Third parties who have an interest in the system but no direct regulatory
- authority
- Society
- Environment

Identifying stakeholders

One way to help identify stakeholders is by answering the following set of questions:

- Who is paying for the system?
- Who is going to use the system?
- Who is going to judge the fitness of the system for use?
- What agencies and entities regulate any aspect of the system?
- What laws govern the construction, deployment, and operation of the system?
- Who is involved in any aspect of the specification, design, construction, testing, maintenance, and retirement of the system?

Stakeholder/User Classes

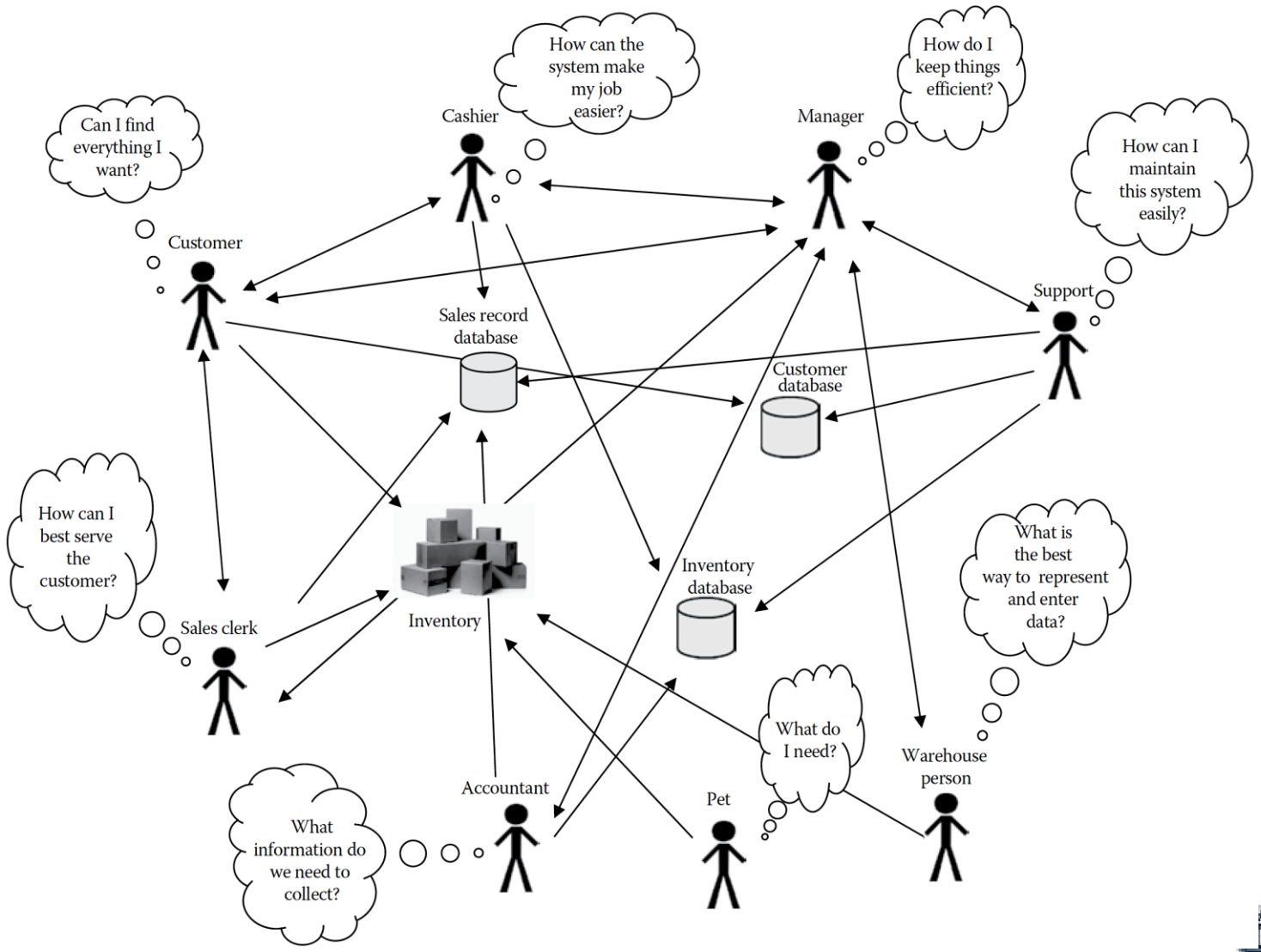
- Once the stakeholder groups have been identified, it may be necessary to divide these groups into classes to adequately address their needs and desires.
- A stakeholder/user class subdivision is usually necessary when the classes are large and/or heterogeneous.
- In many cases, class subdivision will be needed for the collection of system users.

Stakeholder/User Classes

For example, for the pet store POS system user classes would include:

- ✧ Cashiers
- ✧ Managers
- ✧ System maintenance personnel (to make upgrades and fixes)
- ✧ Store customers
- ✧ Inventory/warehouse personnel (to enter inventory data)
- ✧ Accountants (to enter tax information)
- ✧ Sales department (to enter pricing and discounting information)

Rich Pictures for Stakeholder Identification



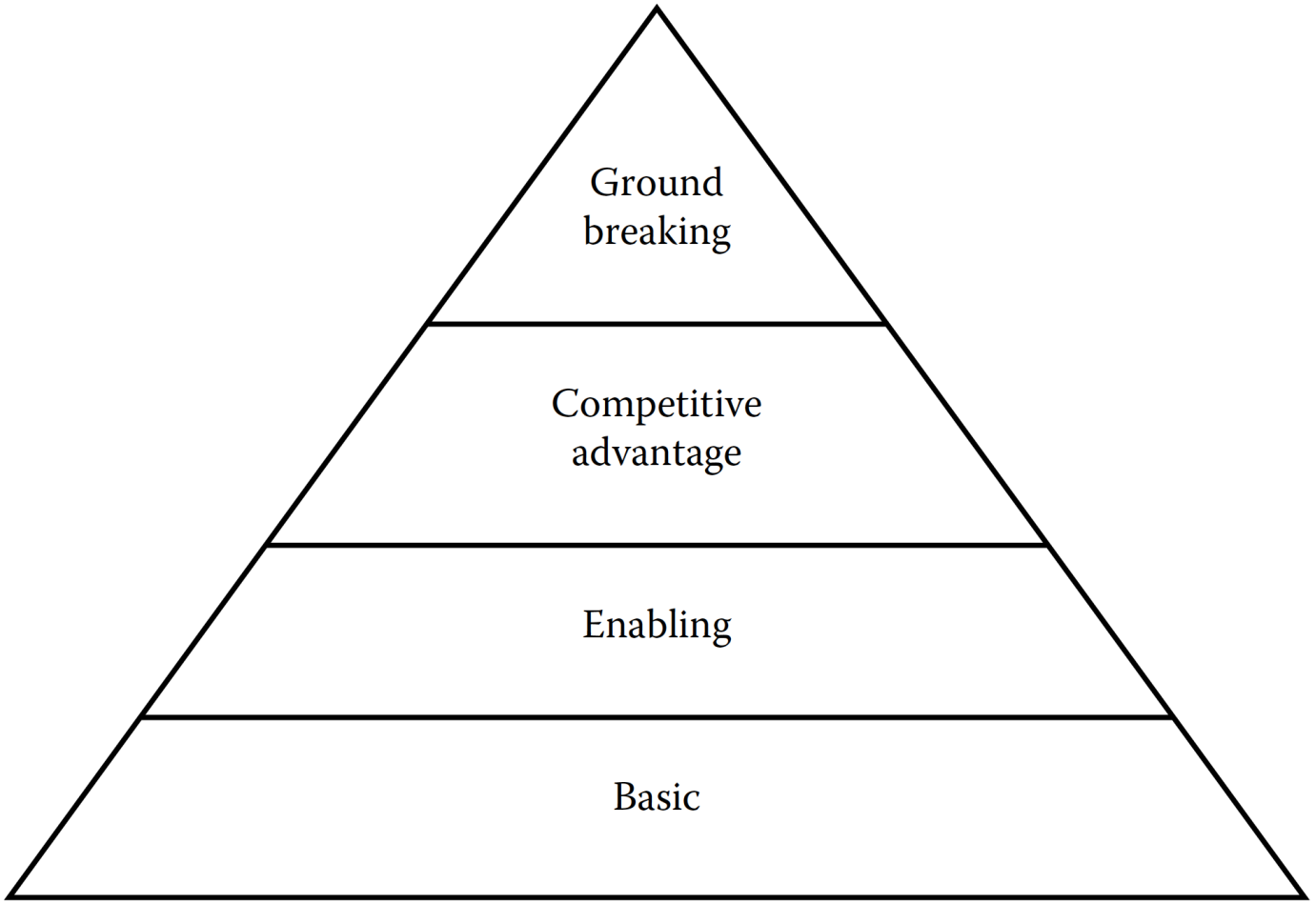
Rich picture for pet store POS system

Customer Wants and Needs

What Do Customers Want?

- The requirements engineer seeks to satisfy customer wants and needs, but it is not always easy to know what these are. Why?
- Because customers' wants and needs exist on many levels
- Requirements engineers have to help customers to set realistic goals for the system to be built

Customer Wants and Needs



Hierarchy of customer needs/wants

Customer Wants and Needs

For the pet store POS system, customers want

- ✧ Speed
- ✧ Accuracy
- ✧ Clarity (in the printed receipt)
- ✧ Efficiency
- ✧ Ease of use (especially if self-service provided)
- ✧ and more

Customer Wants and Needs

What Don't Customers Want?

- Sometimes customers are very explicit in what they don't want the system to do.
- These specific undesirable features or "do not wants" or "shall not" requirements are frequently overlooked by the requirements engineer.
- Unwanted features can include
  - Undesirable performance characteristics
  - Aesthetic features
  - Gold-plating (excessive and unnecessary features)
  - Safety concerns (hazards)

Customer Wants and Needs

Here are some “shall not” requirements for the pet store POS system:

- If the register tape runs out, the system shall not crash.
- If a product code is not found, the system shall not crash.
- If a problem is found in the inventory reconciliation code, the current transaction shall not be aborted.

Customers Change their Minds

One of the greatest challenges in dealing with customers is that they sometimes don't know precisely what they want the system to do. Why?

- A customer can't see every possible desideratum
- Importance might change as these requirements change during the system life cycle
- Sometimes the environment in which the system functions and the customers operate changes
- Due to changes of the return on investment
- Sometimes the customer is simply inconsistent
- Customers will deliberately withhold information for a variety of reasons

Stakeholder Prioritization

- Not all stakeholders are of equal importance.
- Ranking the stakeholders will lead to requirements prioritization, which is the key to reconciliation and risk mitigation

Stakeholder Class	*Rank*	*Rationale*
Cashiers	2	They have the most interaction with the system.
Managers	1	They are the primary customer/sponsor.
System maintenance personnel	4	They have to fix things when they break.
Store customers	3	They are the most likely to be adversely affected.
Inventory/warehouse personnel	6	They have the least direct interaction with the system.
Accountants/sales personnel	5	They read the reports.

Partial Ranking of Stakeholders for the Pet Store POS System

Communicating with Stakeholders

- One of the most important activities of the requirements engineer is to communicate with all stakeholders.
- It is essential that all communications be conducted clearly, ethically, consistently, and in a timely fashion
- What is the best format for communication with stakeholders?
  - Interview
  - Group meeting
  - Seminar
  - Etc.

Interviews in practice

- Normally a mix of closed and open-ended interviewing.
- Interviews are good for getting an overall understanding of what stakeholders do and how they might interact with the system.
- Interviewers need to be open-minded without pre-conceived ideas of what the system should do
- You need to prompt the use to talk about the system by suggesting requirements rather than simply asking them what they want.

Problems with interviews

- Application specialists may use language to describe their work that isn't easy for the requirements engineer to understand.
- Interviews are not good for understanding domain requirements
  - Requirements engineers cannot understand specific domain terminology;
  - Some domain knowledge is so familiar that people find it hard to articulate or think that it isn't worth articulating.

Stories and scenarios

- Scenarios and user stories are real-life examples of how a system can be used.
- Stories and scenarios are a description of how a system may be used for a particular task.
- Because they are based on a practical situation, stakeholders can relate to them and can comment on their situation with respect to the story.

Scenarios

- A structured form of user story
- Scenarios should include
  - A description of the starting situation;
  - A description of the normal flow of events;
  - A description of what can go wrong;
  - Information about other concurrent activities;
  - A description of the state when the scenario finishes.

Stakeholder Negotiations

- Requirements engineer must negotiate with customers and other stakeholders.
- The negotiations deal with convincing the customer that some desired functionality is impossible or too costly
- Expectation setting and management throughout the life cycle of any system project is an exercise in negotiation
- Make sure that the scope and duration of the discussions are agreed
- Trying to eliminate unwanted surprises for both sides
- Understand people's expectations.
- Look for early successes
- Conclude negotiating only when all parties are satisfied

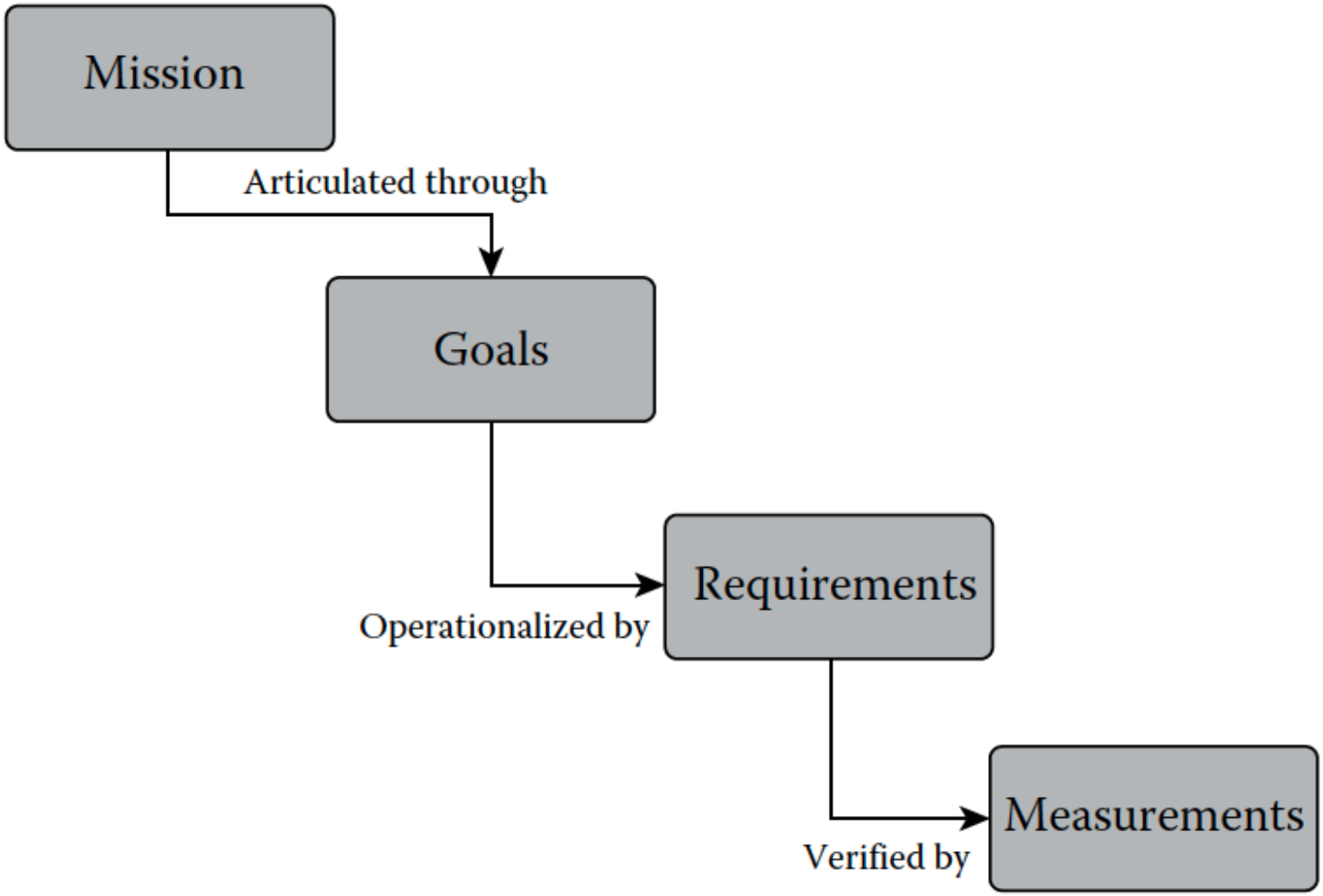
Uncovering Stakeholder Goals

- Successful communications and negotiations will help to uncover and clarify all stakeholders' goals.
- Goals further detail the intentions of the system summarized in the product mission statement
- For example, some goals for the pet store POS:
  - Provide "hassle free" shopping for all customers
  - Support all coupon and discount processing
  - Support all customer loyalty programs
  - Fully automate inventory entry and maintenance
  - Support all local, state, and federal tax processing

Uncovering Stakeholder Goals

- Goals provide further articulation of the intent contained in the product mission statement
- Goals are operationally described and detailed through the requirements, which must be verified through measurements.
- Goal-oriented requirements engineering involves the analysis of stakeholder goals in order to obtain new functional requirements to meet these goals
- These approaches aim at modelling the “who, what, why, where, when, and how” of requirements

Goal-based Requirements Engineering



Requirements specification

- The process of writing down the user and system requirements in a requirements document.
- User requirements have to be understandable by end-users and customers who do not have a technical background.
- System requirements are more detailed requirements and may include more technical information.
- The requirements may be part of a contract for the system development
  - It is therefore important that these are as complete as possible.

Ways of writing a system requirements specification

Notation	Description
Natural language	The requirements are written using numbered sentences in natural language. Each sentence should express one requirement.
Structured natural language	The requirements are written in natural language on a standard form or template. Each field provides information about an aspect of the requirement.
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 now rarely used although it can be useful for interface specifications.
Graphical notations	Graphical models, supplemented by text annotations, are used to define the functional requirements for the system; UML use case and sequence diagrams are commonly used.
Mathematical specifications	These notations are based on mathematical concepts such as finite-state machines or sets. Although these unambiguous specifications can reduce the ambiguity in a requirements document, most customers don't understand a formal specification. They cannot check that it represents what they want and are reluctant to accept it as a system contract

Natural language specification

- Requirements are written as natural language sentences supplemented by diagrams and tables.
- Used for writing requirements because it is expressive, intuitive and universal. This means that the requirements can be understood by users and customers.

Guidelines for writing requirements

- Invent 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 the use of computer jargon.
- Include an explanation (rationale) of why a requirement is necessary.

Problems with natural language

- ✧ Lack of clarity
  - Precision is difficult without making the document difficult to read.
- ✧ Requirements confusion
  - Functional and non-functional requirements tend to be mixed-up.
- ✧ Requirements amalgamation
  - Several different requirements may be expressed together.

Structured specifications

- An approach to writing requirements where the freedom of the requirements writer is limited and requirements are written in a standard way.
- This works well for some types of requirements e.g. requirements for embedded control system but is sometimes too rigid for writing business system requirements.

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.
- Information needed for the computation and other entities used.
- Description of the action to be taken.
- Pre and post conditions (if appropriate).
- The side effects (if any) of the function.

Tabular specification

- Used to supplement natural language.
- Particularly useful when you have to define a number of possible alternative courses of action.
- For example, the insulin pump systems bases its computations on the rate of change of blood sugar level and the tabular specification explains how to calculate the insulin requirement for different scenarios.

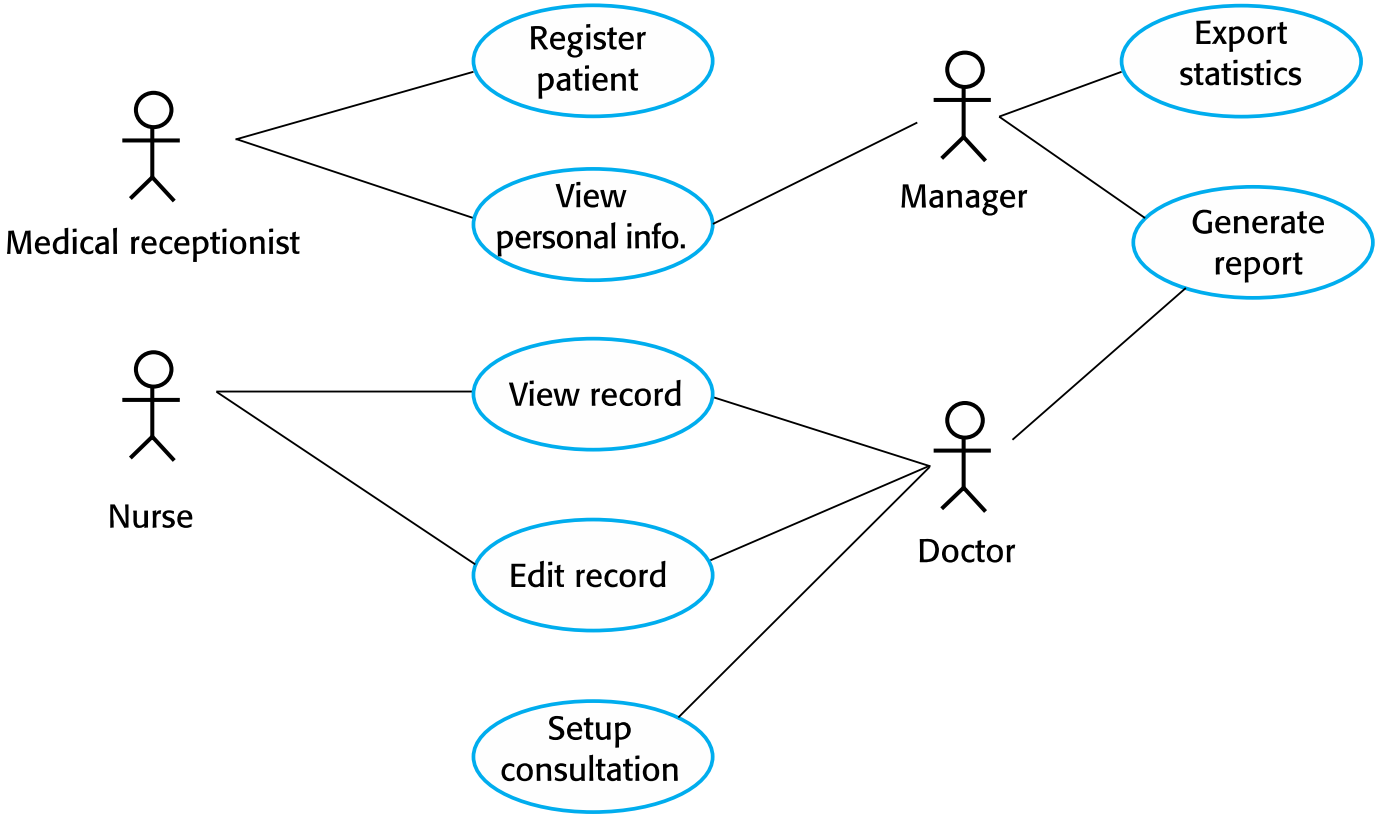
Model-based Specification

- FSM
- DFD
- UML
- PN, CPN
- Formal specification
- Mathematical system model
- Z specification language

Use cases (graph based)

- Use-cases are a kind of scenario that are included in the UML.
- Use cases identify the actors in an interaction.
- A set of use cases should describe all possible interactions with the system.
- High-level graphical model supplemented by more detailed tabular description
- UML sequence diagrams may be used to add detail to use-cases by showing the sequence of event processing in the system.

Use cases for the Mentcare system



The software requirements document

- The software requirements document is the official statement of what is required of the system developers.
- Should include both a definition of user requirements and a specification of the system requirements.
- It is NOT a design document. As far as possible, it should set of WHAT the system should do rather than HOW it should do it.

Requirements document variability

- Information in requirements document depends on type of system and the approach to development used.
- Systems developed incrementally will, typically, have less detail in the requirements document.
- Requirements documents standards have been designed e.g. IEEE standard. These are mostly applicable to the requirements for large systems engineering projects.

The structure of a requirements document

Chapter	Description
Preface	This should define the expected readership of the document and describe its version history, including a rationale for the creation of a new version and a summary of the changes made in each version.
Introduction	This should describe the need for the system. It should briefly describe the system's functions and explain how it will work with other systems. It should also describe how the system fits into the overall business or strategic objectives of the organization commissioning the software.
Glossary	This should define the technical terms used in the document. You should not make assumptions about the experience or expertise of the reader.
User requirements definition	Here, you describe the services provided for the user. The nonfunctional system requirements should also be described in this section. This description may use natural language, diagrams, or other notations that are understandable to customers. Product and process standards that must be followed should be specified.
System architecture	This chapter should present a high-level overview of the anticipated system architecture, showing the distribution of functions across system modules. Architectural components that are reused should be highlighted.

The structure of a requirements document

Chapter	Description
System requirements specification	This should describe the functional and nonfunctional requirements in more detail. If necessary, further detail may also be added to the nonfunctional requirements. Interfaces to other systems may be defined.
System models	This might include graphical system models showing the relationships between the system components and the system and its environment. Examples of possible models are object models, data-flow models, or semantic data models.
System evolution	This should describe the fundamental assumptions on which the system is based, and any anticipated changes due to hardware evolution, changing user needs, and so on. This section is useful for system designers as it may help them avoid design decisions that would constrain likely future changes to the system.
Appendices	These should provide detailed, specific information that is related to the application being developed; for example, hardware and database descriptions. Hardware requirements define the minimal and optimal configurations for the system. Database requirements define the logical organization of the data used by the system and the relationships between data.
Index	Several indexes to the document may be included. As well as a normal alphabetic index, there may be an index of diagrams, an index of functions, and so on.

Requirements validation

- Concerned with whether the requirements satifisfy customers' needs.
- Requirements error costs are high so validation is very important
  - Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error.

Requirements checking

- ✧ Validity. Does the system provide the functions which best support the customer's needs?
- ✧ Consistency. Are there any requirements conflicts?
- ✧ Completeness. Are all functions required by the customer included?
- ✧ Realism. Can the requirements be implemented given available budget and technology
- ✧ Verifiability. Can the requirements be checked?

Requirements validation techniques

- Requirements reviews
  - Systematic manual analysis of the requirements.
- Prototyping
  - Using an executable model of the system to check requirements.
- Test-case generation
  - Developing tests for requirements to check testability.

Requirements management

- Requirements management is the process of managing changing requirements during the requirements engineering process and system development.
- New requirements emerge as a system is being developed and after it has gone into use.
- You need to keep track of individual requirements and maintain links between dependent requirements so that you can assess the impact of requirements changes.
- You need to establish a formal process for making change proposals and linking these to system requirements.

Requirements management planning

- Establishes the level of requirements management detail that is required.
- Requirements management decisions:
  - *Requirements identification* Each requirement must be uniquely identified so that it can be cross-referenced with other requirements.
  - *A change management process* This is the set of activities that assess the impact and cost of changes.
  - *Traceability policies* These policies define the relationships between each requirement and between the requirements and the system design that should be recorded.
  - *Tool support* Tools that may be used range from specialist requirements management systems to spreadsheets and simple database systems.

Requirements change management

✧ Deciding if a requirements change should be accepted

- *Problem analysis and change specification*
  - During this stage, the problem or the 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 request.
- *Change analysis and costing*
  - The effect of the proposed change is assessed using traceability information and general knowledge of the system requirements. Once this analysis is completed, a decision is made whether or not to proceed with the requirements change.
- Change implementation
  - The requirements document and, where necessary, the system design and implementation, are modified. Ideally, the document should be organized so that changes can be easily implemented.

Requirements change management