LECTURE
REQUIREMENTS ENGINEERING

Waris Ali

Lecture Outcomes

At the end of this lecture you should be able to answer the following questions:

1. What are the seven stages in requirements engineering?
2. Can you describe each of these stages?
3. Name the different models created in the elaboration phase?
4. Why is it necessary to validate the requirements?
5. What are the different areas that are uncovered during validation?

Requirements Engineering

“The hardest single part of building a software system is deciding what to build. No part of the work so cripples the resulting system if done wrong. No other part is difficult to rectify later.”

Fred Brooks

Requirements Engineering

- Challenge facing system and software engineers
  - How can we ensure that we have specified a system that:
    - Properly meets the customer's needs
    - Satisfies the customer's expectations
- Requirements engineering provides mechanisms for:
  - Understanding what the customer wants
  - analyzing need
  - assessing feasibility
  - negotiating a reasonable solution
  - specifying the solution
  - validating the specification
  - managing the transformation of the requirements into an operational system

Requirements Engineering-I

- Inception—Ask a set of questions that establish …
  - Basic understanding of the problem
  - The people who want a solution
  - The nature of the solution that is desired
  - The effectiveness of preliminary communication and collaboration between the customer and the 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-II

- Specification—Can be any one (or more) of the following:
  - A written document
  - A set of models
  - A formal mathematical
  - A collection of user scenarios (use-cases)
  - A prototype
- 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 (unachievable) requirements.

Requirements Engineering-III

- Requirements management
  - Features traceability table
    - Requirement related to customer system/product features
  - Source traceability table
    - Identify sources related to requirements
  - Dependency traceability table
    - Identify the dependencies of requirements on other requirements
  - Subsystem traceability table
    - Categorizes requirements on basis of subsystem
  - Interface traceability table
    - Requirement relate to both internal and external system interfaces

Chinese Proverb

“He who ask a question is a fool for five minutes; he who does not ask a question is a fool forever”

Inception

- Identify stakeholders
  - “who else do you think I should talk to?”
- Recognize multiple points of view, Ask the customer, users, and others:
  - What the objectives for the system are
  - What is to be accomplished
  - How the system fits the into the needs of the business
  - How the system will be used on a day-to-day basis

Inception

- Work toward collaboration
- The first questions
  - Who is behind the request for this work?
  - Who will use the solution?
  - What will be the economic benefit of a successful solution
  - Is there another source for the solution that you need?

Requirement Elicitation

It might seem that this should be simple:

- Meetings are conducted and attended by both software engineers and customers
- Rules for preparation and participation are established
- An agenda is suggested
- A "facilitator" (can be a customer, a developer, or an outsider) controls the meeting
- A "definition mechanism" (can be work sheets, flip charts, or wall stickers or an electronic bulletin board, chat room or virtual forum) is used
- The goal is
  - To identify the problem
  - Propose elements of the solution
  - Negotiate different approaches, and
  - Specify a preliminary set of solution requirements

Requirement Elicitation

In fact it is hard!

- Problems of scope
- Problems of understanding
- Problems of volatility

Problems of Scope

- The boundary of the system may be ill-defined
  - Who is going to interact with the system?
  - What other systems are involved?
  - Exactly what functionality is the responsibility of the system
    - e.g. should a rostering system produce a telephone directory?
- The customer/user may specify unnecessary technical detail that may confuse overall system objectives
  - e.g. specifying OS, language, hardware, etc. for no particularly good reason

Problems of Understanding

- Customers/users may:
  - Not be completely sure of what is needed, e.g.:
    - “See what you can do to help us”
      (Marketing director of textile business)
    - “Try to improve the project”
      (Director of British Aircraft Corporation, with reference to the Concorde project)
  - Have a poor understanding of the capabilities and limitations of their computing equipment
  - Not have a full understanding of the problem domain
  - Have trouble communicating needs to system engineer
  - Omit information believed to be “obvious”
  - Specify requirements that conflict with needs of others
  - Specify requirements that are ambiguous or untestable

Problems of Volatility

- Requirements change over time
- Change is inevitable in most systems, due to factors such as:
  - Changes in customer organization, e.g.
    - new divisions, new products
  - Changes in scale, e.g.
    - Number of transactions per day
    - Number of users
    - Increased connection bandwidth
  - External changes
    - Changes in law (e.g. taxation)
    - Changes in international standards (e.g. MPEG)
  - Customers get new ideas as they become aware of system possibilities

Overcoming Requirements Elicitation Problems

- Sommerville and Sawyer give guidelines for addressing these problems
  - Assess business and technical feasibility for the proposed system
  - Identify the people who will help to specify requirements, and understand their organizational bias
  - Define the technical environment in which the system will be placed:
    - Computing architecture, operating system, telecommunications needs
  - Identify "domain constraints" that limit system functionality or performance
    - Characteristics of business environment specific to application domain

Overcoming Requirements Elicitation Problems (cont.)

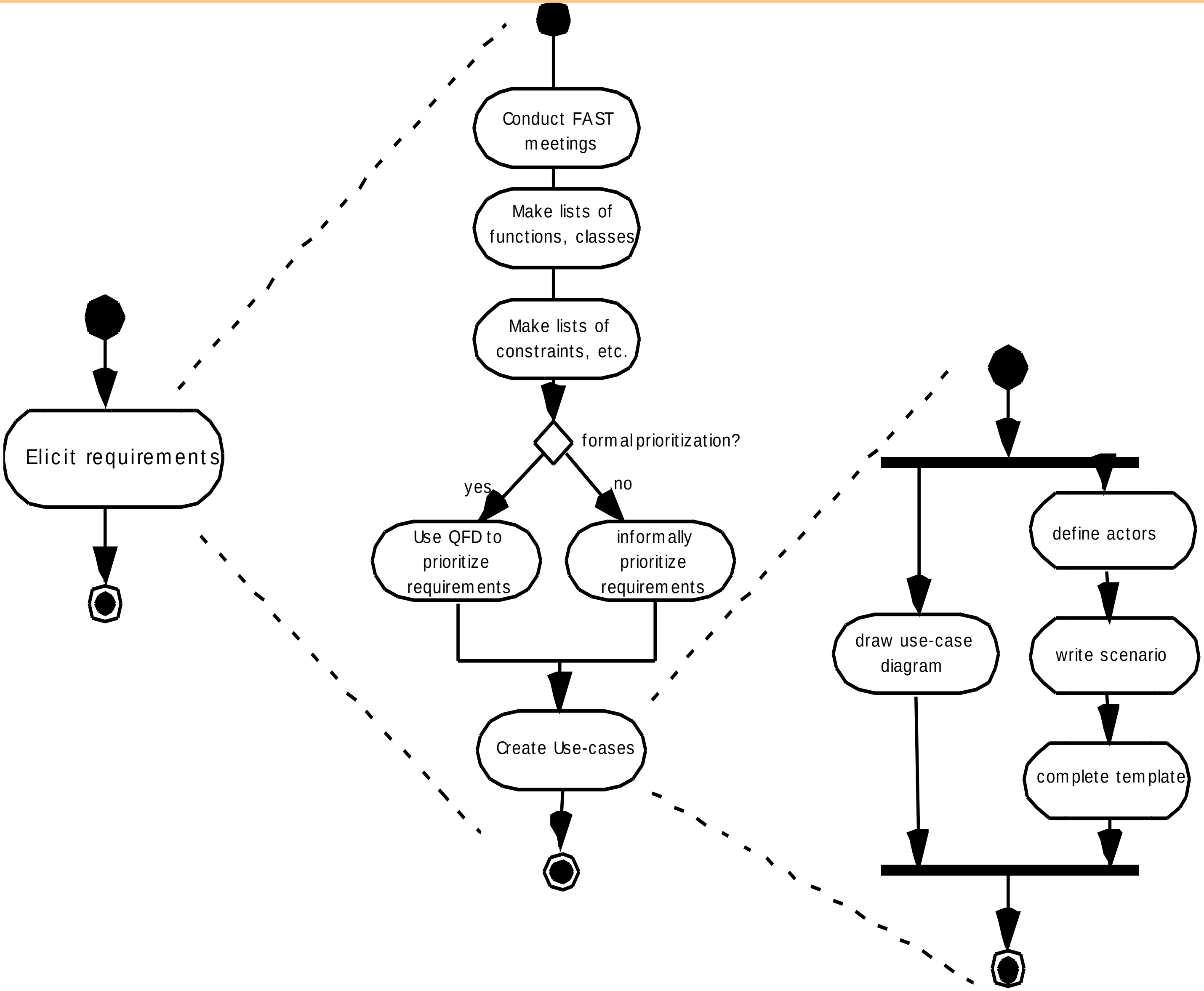
- Define one or more requirements elicitation methods
  - Interviews, focus groups, team meetings
- Ensure that many people participate so that requirements are defined from different points of view
  - Identify and record the rationale for each requirement
- Identify ambiguous requirements as candidates for prototyping
  - A means of addressing volatility
- Create usage scenarios to help customers and users better identify key requirements

Elicitation Work Products

- A statement of need and feasibility
- A bounded statement of scope for the system or product.
- A list of customers, users, and other stakeholders who participated in requirements elicitation
- A description of the system's technical environment.
- A list of requirements (preferably organized by function) and the domain constraints that apply to each
- A set of usage scenarios that provide insight into the use of the system or product under different operating conditions
- Any prototypes developed to better define requirements

All work products are reviewed by participants in requirements elicitation

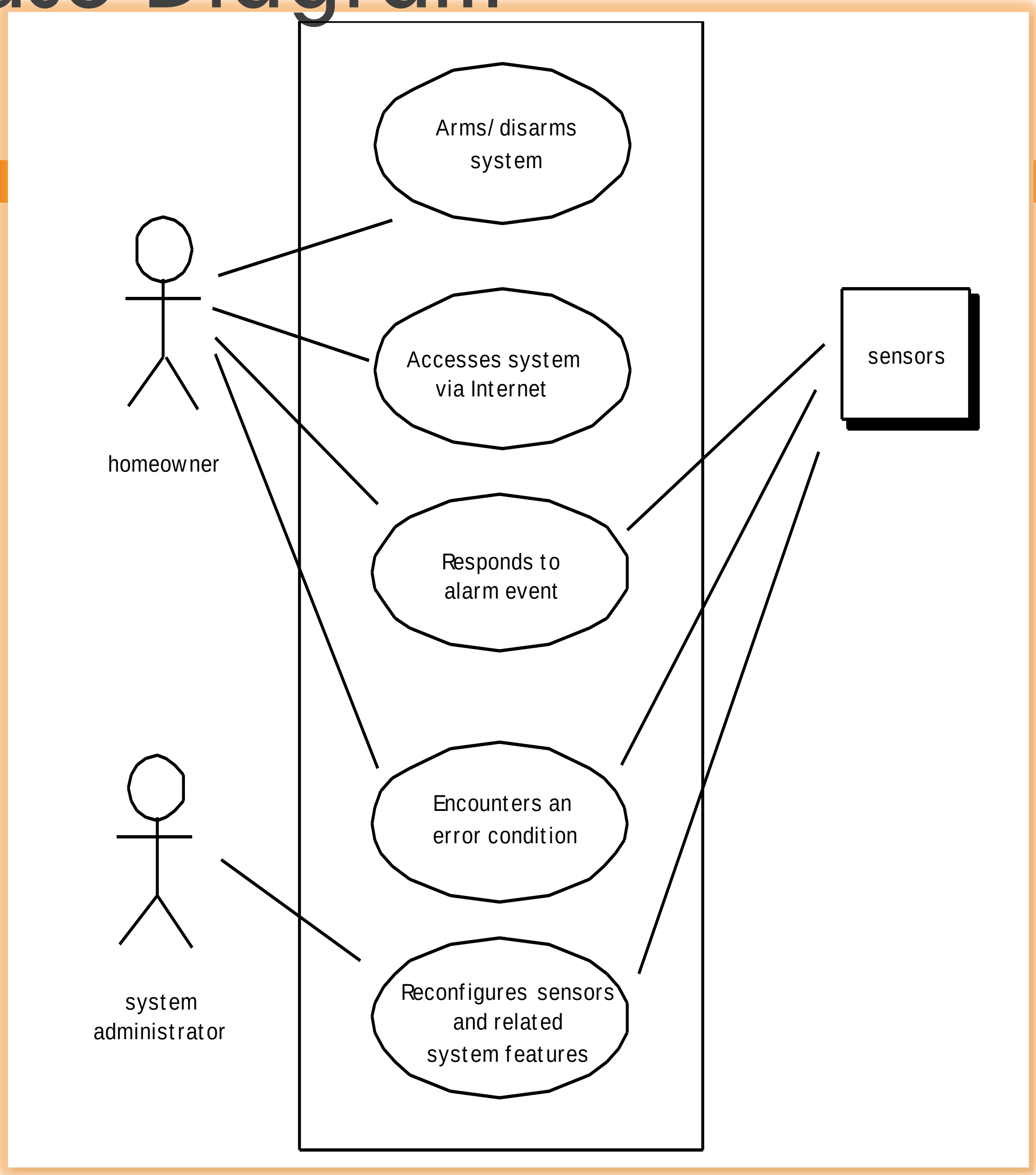
Eliciting Requirements



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?
  - 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


Arms/ disarms system
Accesses system via Internet
Responds to alarm event
Encounters an error condition
Reconfigures sensors and related system features
homeowner
system administrator
sensors

Requirements Analysis

- The products of requirements elicitation form the basis for *requirements analysis*
- Requirements analysis
  - Categorizes requirements
    - Organizes them into related sub-sets
  - Explores relationships between requirements
  - Examines requirements for
    - Consistency
    - Omissions
    - Ambiguity
  - Prioritizes requirements based on customer/user needs
    - May lead to plan for incremental development

Requirements Analysis Questions

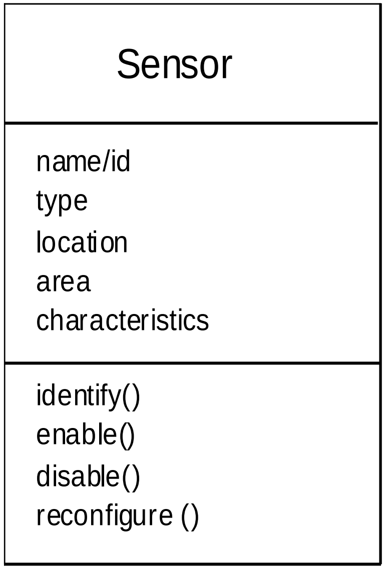
- Is each requirement consistent with the overall objective for the system?
- Have all requirements been specified at the proper level of abstraction?
  - i.e. not too much technical detail, or exclusion of future possibilities
- Is each requirement really necessary?
  - Or is it an add-on not needed for core system objective?
- Is each requirement bounded an unambiguous?
- Does each requirement have an attribution?
  - Who or what is the source of the requirement?
- Are there any conflicting requirements?
- Is each requirement technically achievable in specified environment?
- Is each requirement testable once implemented?

Building the 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 …



Sensor

name/id
type
location
area
characteristics

identify()
enable()
disable()
reconfigure ()

Analysis Patterns

Pattern name: A descriptor that captures the essence of the pattern.

Intent: Describes what the pattern accomplishes or represents

Motivation: A scenario that illustrates how the pattern can be used to address the problem.

Forces and context: A description of external issues (forces) that can affect how the pattern is used and also the external issues that will be resolved when the pattern is applied.

Solution: A description of how the pattern is applied to solve the problem with an emphasis on structural and behavioral issues.

Consequences: Addresses what happens when the pattern is applied and what trade-offs exist during its application.

Design: Discusses how the analysis pattern can be achieved through the use of known design patterns.

Known uses: Examples of uses within actual systems.

Related patterns: On e or more analysis patterns that are related to the named pattern because (1) it is commonly used with the named pattern; (2) it is structurally similar to the named pattern; (3) it is a variation of the named pattern.

Negotiation

“A compromise is the art of dividing a cake in such a way that everyone believes he has the biggest piece” Ludwig Erhard

Requirements Negotiation

- It is common for customers/users to ask for more than can be achieved
- Also common for different stakeholders to proposed conflicting requirements
  - e.g. cost requirement from management vs. performance requirement from users
  - Each party might argue that their requirement is "essential"

Negotiating Requirements

Requirements engineer must resolve these conflicts through negotiation:

- Identify the key stakeholders
  - These are the people who will be involved in the negotiation
- Determine each of the stakeholders “win conditions”
  - Win conditions are not always obvious
- Negotiate
  - Work toward a set of requirements that lead to “win-win”

In an iterative process, modify, combine or eliminate requirements so that each stakeholder gets some satisfaction

Validating Requirements-I

- 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 may not be essential to the objective of the system?
- Is each requirement bounded and unambiguous?
- Does each requirement have attribution? That is, is a source (generally, a specific individual) noted for each requirement?
- Do any requirements conflict with other requirements?

Validating Requirements-II

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

Requirements Management

- We have noted that changes in requirements:
  - are essentially unavoidable
  - will persist throughout the lifetime of the system
- *Requirements management* helps the project team to identify, track and control requirements and changes to them
  - This is closely related to configuration management
- Traceability tables are developed for requirements

Traceability Tables

- Features traceability table
  - Show how requirements relate to customer-observable system features
- Source traceability table
  - Identify the source of each requirement
- Dependency traceability table
  - Show relationships between requirements
- Subsystem traceability table
  - Categorise requirements according to subsystems they govern
- Interface traceability table
  - Shows how requirements relate to internal and external system interfaces

QUESTIONS ?