CHAPTER

3

Software Requirements

Learning Outcomes

In this chapter you will learn:

- Basic terminology in software requirements
- Product vision and project scope document
- Difference between functional and non-functional requirements (NFRs)
- Types of NFRs
- Requirements engineering process steps
- Techniques for requirements elicitation
- Development of a use case model
- Validation of software requirements
- IEEE standard for writing a software requirements specifications document

Software analysis

- What the software under development is supposed to do?
  - Understand the scope
  - Capturing user’s requirements
  - Identifying constraints
  - Understanding interfaces with entities, users and external systems
- Should be design independent and detailed enough
- Validated by client and developers

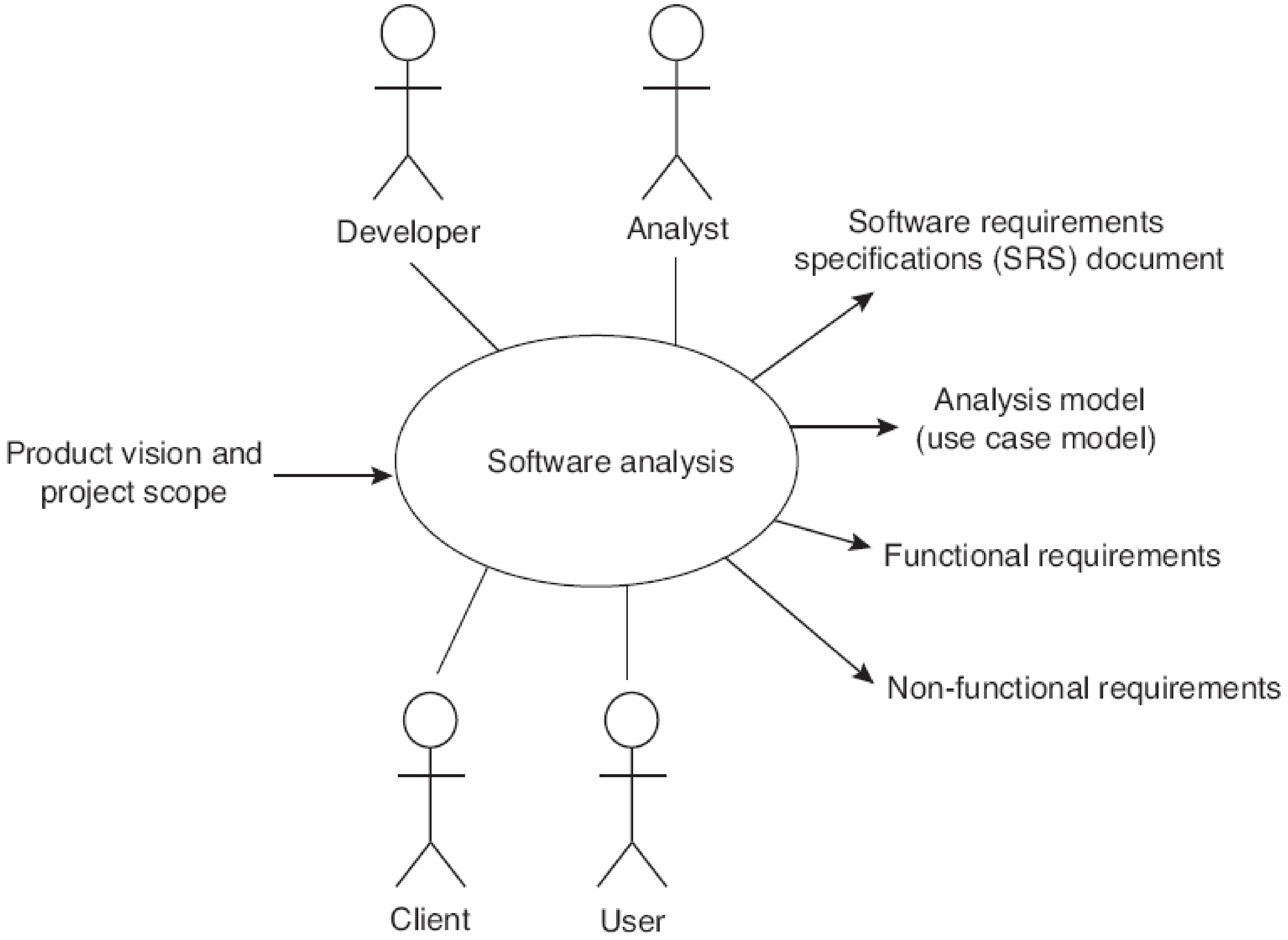
Developer
Analyst
Software requirements specifications (SRS) document
Analysis model (use case model)
Product vision and project scope
Software analysis
Functional requirements
Non-functional requirements
Client
User

Figure 3.1 Inputs to and outputs from the software analysis phase

Requirements

- First step in the analysis phase
- Functional and non-functional requirements
- Requirements engineering process starting from product vision and project scope
- Use case model approach for capturing user requirements

Product vision and project scope

- Establish business requirements and aligning them with software requirements specs
  - Clear vision and scope lead to more focused requirements
- Document contains 3 sections:
  - Business requirements and context
  - Product vision
  - Project scope and limitations

Table 3.1 Template for a software product vision and project scope document

Title page
Change history
Preface
Table of contents
1. Business requirements and context
    1.1 Product rationale and business opportunity
    1.2 Business objectives
    1.3 Business risks
    1.4 Stakeholders
    1.5 Context and environment
2. Product vision
    2.1 Statement of product goals
    2.2 Main features
    2.3 Assumptions and dependencies
3. Project scope
    3.1 Scope of current release
    3.2 Scope of future releases
    3.3 Limitations
Annexes
Index

Business requirements and context

- Brief description of the rationale and business opportunity for the product to be developed.
- Main business objectives and some quantifiable success criteria for these objectives.
- Business objectives can refer to quantifiable financial or technical benefits to the target customers that are preferably quantifiable.
- A quantifiable financial benefit is: “The product helps reducing the customer support cost by 30% during the first year of operations”.
- A quantifiable technical benefit is: “The average delay to deal with and service a customer complaint will be around 8 hours”.

Business requirements and context (2)

- Business risks associated with the business requirements and their management.
  - Risk assessment, prioritization, monitoring, mitigation and control. Detailed elaboration on the technical and non-technical project risks will be included in the software project plan document as will be discussed later in Chapter 10.
- Description of the different types of product users and other stakeholders.
- Context and environment in which the product will be operating.
  - The operating context and environment will later determine many of the non-functional product quality requirements such as security, performance, availability, and business continuity requirements.

Product vision

- Clear statement of the product goals.
  - Capture the views of possibly many stakeholders and types of users, and show the financial and non-financial business benefits
- Identify main functionalities and product features
- List all the assumptions and dependencies
  - help identifying and reducing the risks, and clarifying constraints to various project and product stakeholders.
  - external conditions that have to be met: e.g. availability of a hardware device to interface with.

Project scope and limitations

- List functionalities and features to be included in different releases or versions of the product.
- Scope for each release is developed in line with the stakeholders' objectives and priorities.
  - eliminating future misunderstandings between developers and clients.
- Limitations include the functionalities and features that will not be provided by the product in any of its future releases.
  - clarify the expectations and functional boundaries of the product
  - reduce possible future conflicts between various product stakeholders.

Requirements engineering

- Bad requirements => bad product
- Need for a formal and structured process for developing and managing requirements
- Requirements development (engineering):
  - requirements elicitation: techniques for seeking, extracting, acquiring and documenting requirements
  - requirements analysis
  - requirements specification
  - requirements maintenance (part of configuration management)

Requirements management

- Part of software project management issues
- Proper management of requirements:
  - Prioritization of requirements
  - Requirements change management
  - Negotiation of requirements
  - Quality assurance of requirements

Requirements management activities

- Defining the requirements development processes
- Determining outcomes and deadlines of the requirements engineering phase
- Negotiating and finalizing the software requirements
- Prioritizing the requirements and evaluating the risks
- Tracking and controlling requirements development steps

Requirements management activities (2)

- Acquiring and training on req’s management tools
- Managing the requirements library and controlling its use and defining a change management mechanism
- Determining and collecting key metrics statistics
- Ensuring the quality of the software requirements

Reasons for bad requirements

- Inexistence of a formal process by which the requirements are collected or elicited
- Serious breakdown in the personal communications between the different requirements stakeholders, and mainly between the client and user representatives and the analyst
- Inexistence or lack of a formal requirements validation process
- Failure in the management of the requirements engineering processes
- Lack of use of tools to deal with requirements for large software projects
- Lack of application domain experience of the requirements development team members

Types of requirements

- Functional requirements
  - Functionalities and services provided to users to meet their needs and to achieve business objectives
  - Tangible, visible
- Non-functional requirements
  - Constraints on the provision of functional req’s
  - Quality requirements: system wide or function specific

Elicitation of functional requirements

- Understand client’s needs and communicate them to developers
- Extraction, discovery / invention, acquisition / elaboration of stakeholders needs
- System analyst, business analyst, requirement engineer or requirement facilitator
  - Under and over specification of requirements
  - Understand application domain, identify sources of requirements and stakeholders, selection of elicitation techniques

Elicitation techniques

- Interviews
- Questionnaires
- Task analysis and scenarios
- Preliminary requirements domain analysis
- Brainstorming
- Workshops
- Meetings
- Prototyping
- Ethnographic assessment: describe what is being done by observing or practicing

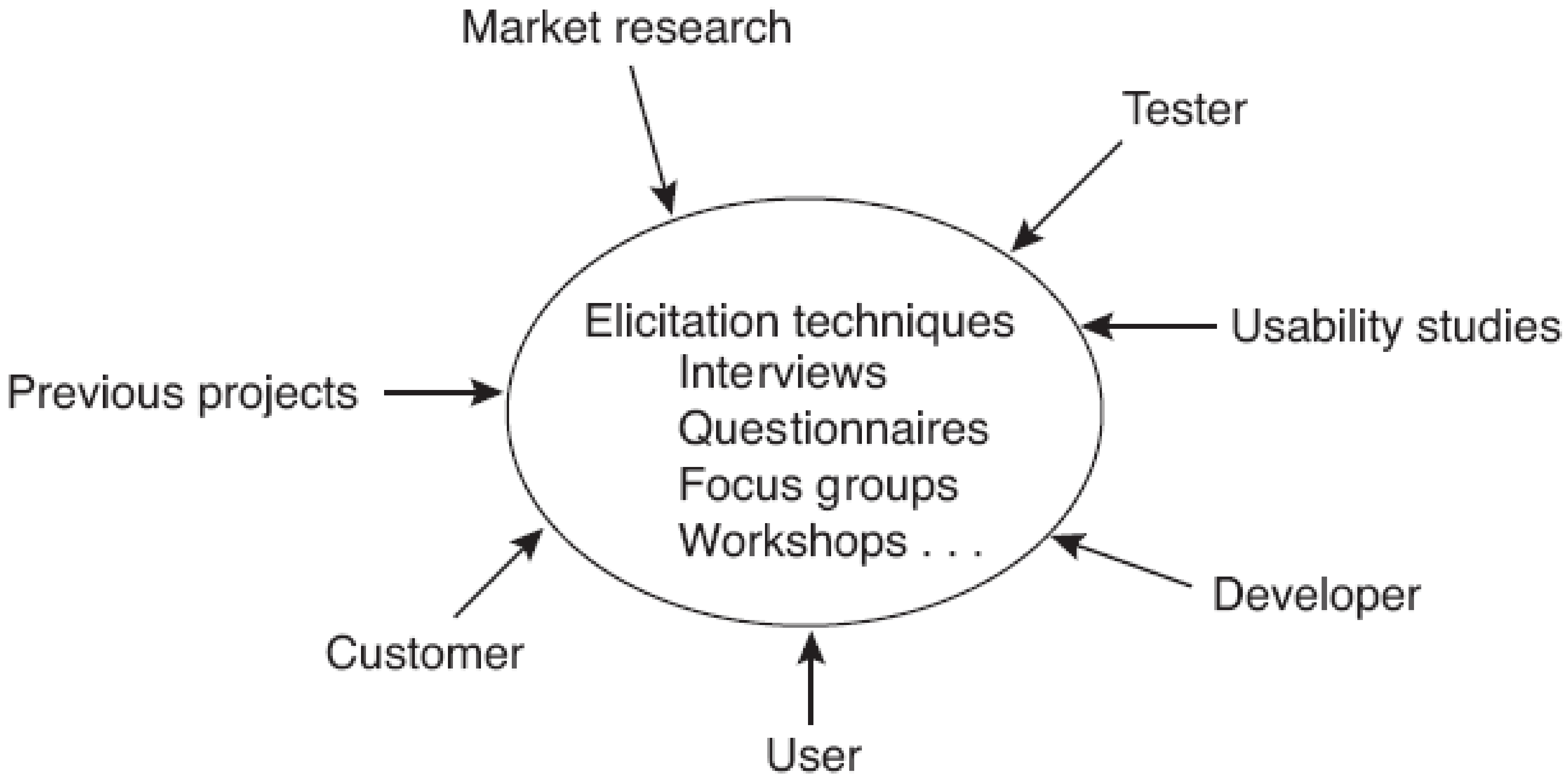

Figure 3.2 Requirements elicitation techniques

Interview technique

- Informal technique used at the beginning of the elicitation process to engage the various participants.
- Individual interviews are used to establish the initial mission statement and goals of the system being developed.
- Interviews can be structured, unstructured, or hybrid.
- **Structured** - requires a set of questions prepared ahead of time and distributed to the participants before the interview.
- **Unstructured** - questions are developed dynamically as the interview progresses.
- **Hybrid** - formatted with a set of fixed questions and impromptu questions that are asked during the interviews.
- Careful formulation of interview questions is needed to avoid negative impact on the participants.

Questionnaire technique

- Used at the beginning of the elicitation process to poll the participants and stakeholders and to obtain their thoughts on issues related to the system.
- Questions must be clear, concise, and appropriate to the issues of the project
  - multiple choice, true and false, or open-ended questions.
- Answers can be used by the requirements engineer to identify initial conflicting requirements requests that need to be resolved later.

Task analysis technique

- Identification of the system uses and the description of the interactions between the users and the system.
- Interactions can be described using scenarios or flow of events.
- Top-down analysis:
  - first high level system tasks are identified.
  - then refined into lower level tasks and sub-tasks for which scenarios have to be elicited.
- Task analysis requires domain knowledge expertise and an understanding of the tasks performed by similar systems.

Preliminary requirements

- Preliminary requirements and domain analysis techniques start from an initial set of requirements.
- Preliminary requirements are proposed by the expert requirements engineer and are passed to the various stakeholders for discussion.
- The engineer must be an authority in the domain of application.

Group meeting technique

- A group meeting includes all participants and stakeholders to start the elicitation process.
- The formation of the group and the dynamics of the interactions among group participants are crucial for the success of this technique.
- Cohesiveness, openness, and transparency are key communications features during group meetings.

Brainstorming technique

- Based on brainstorming sessions: informal meetings to discuss the desirable functionalities and services
- Inventive ideas that might or might not be feasible are generated.
- All ideas are encouraged and the merits are discussed by the requirements analyst and other participants later in more formal settings.
- The end product is an initial set of software requirements.

Workshop technique

- Formal settings used to elicit software requirements.
- Participants -a facilitator, a technical writer, the requirements engineer, and representatives from the client and users’ groups.
- The aim is to reach an agreement on a working draft of the soft ware requirements.
- Must be prepared thoroughly ahead of time and expectations from each participant must be explicitly
- known.
- Elicit best case and worst case scenarios and fit them in the various use cases of the use case model.

Ethnographic assessment technique

- Used by the requirements engineer to gain a greater understanding of the problem domain.
- It relies on the active or passive participation of the engineer in the working of similar systems and their user interfaces.
  - Active: by learning from the users' activities by going through the various scenarios and business flows.
  - Passive: by observing the users' activities without any interference.
- Engineer acts as the apprentice and the user acts as the master or supervisor.

Prototyping technique

- Considered an effective tool for eliciting and validating requirements. The requirements engineer develops an initial prototype of the system. A prototype should show the main elements of the system functionalities.
- A prototype must be easy to develop and modify with little investment. The prototype must be evaluated and modified interactively with the users' input.
- Experience shows that prototype-based requirements are faster to elicit and include fewer errors than requirements developed using other elicitation techniques.

Use case modeling

- Eliciting functional requirements by constructing a use case based requirements model.
- The identification of the use case model is performed using various elicitation techniques.
  - Elicitation of actors, scenarios, use cases and their relationships
  - Use case diagrams (UCD) are developed.

Use case diagram

- Actors – someone or something interacting with system and expects some useful results from it
  - Represented by a stickman
  - Entities external to system
  - People
  - Primary (triggering) and secondary actors
- Use case satisfy some FRs and NFRs
  - Represented by an oval
  - Includes steps to be followed

Use Case Diagram relationships

- A UCD is made of actors, use cases and interrelationships among actors and use cases:
  - <<communicates>>
  - <<extend>>
  - <<include>>
  - generalization (inheritance)

Relationships

- **<<communicate>>** - linking a use case and an actor – represents interactions
- **<<extend>>** -linking use cases – represents one use case interrupted by another user case under certain conditions
- **<<include>>** - linking use cases – represents the inclusion of the steps of one use case in the other user case steps
- **Generalization** or **inheritance** relationship among actors and among use cases (type-of relationship)

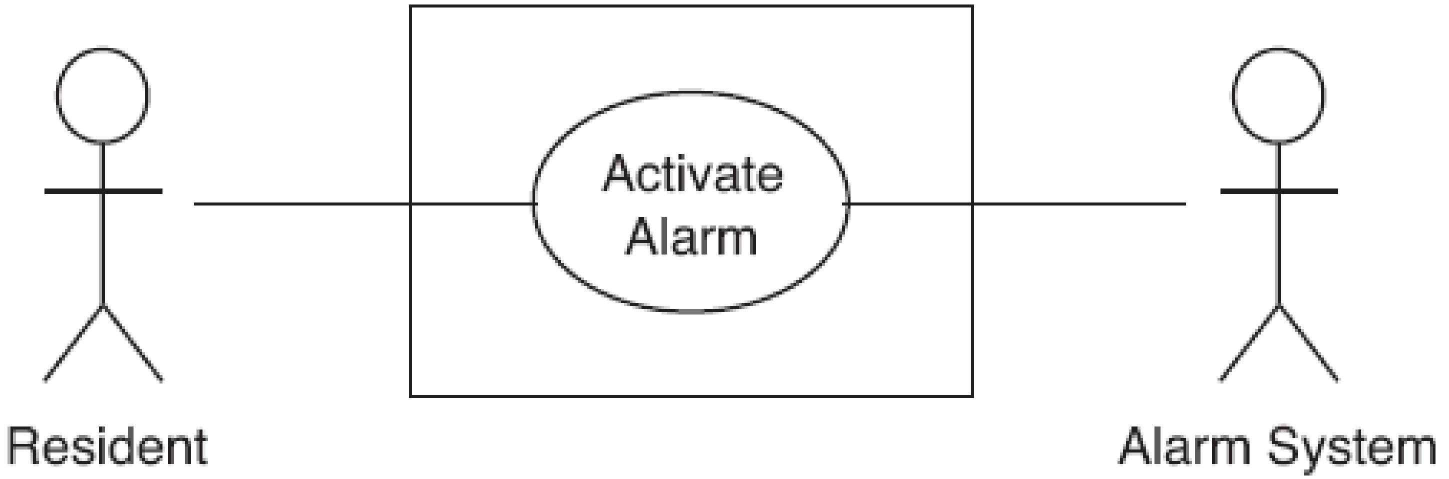

Figure 3.3 Actors Resident and Alarm System and use case Activate Alarm

Actors: Resident (primary actor) and Alarm System
Use case: Activate Alarm
Two <<communicate>> relationships

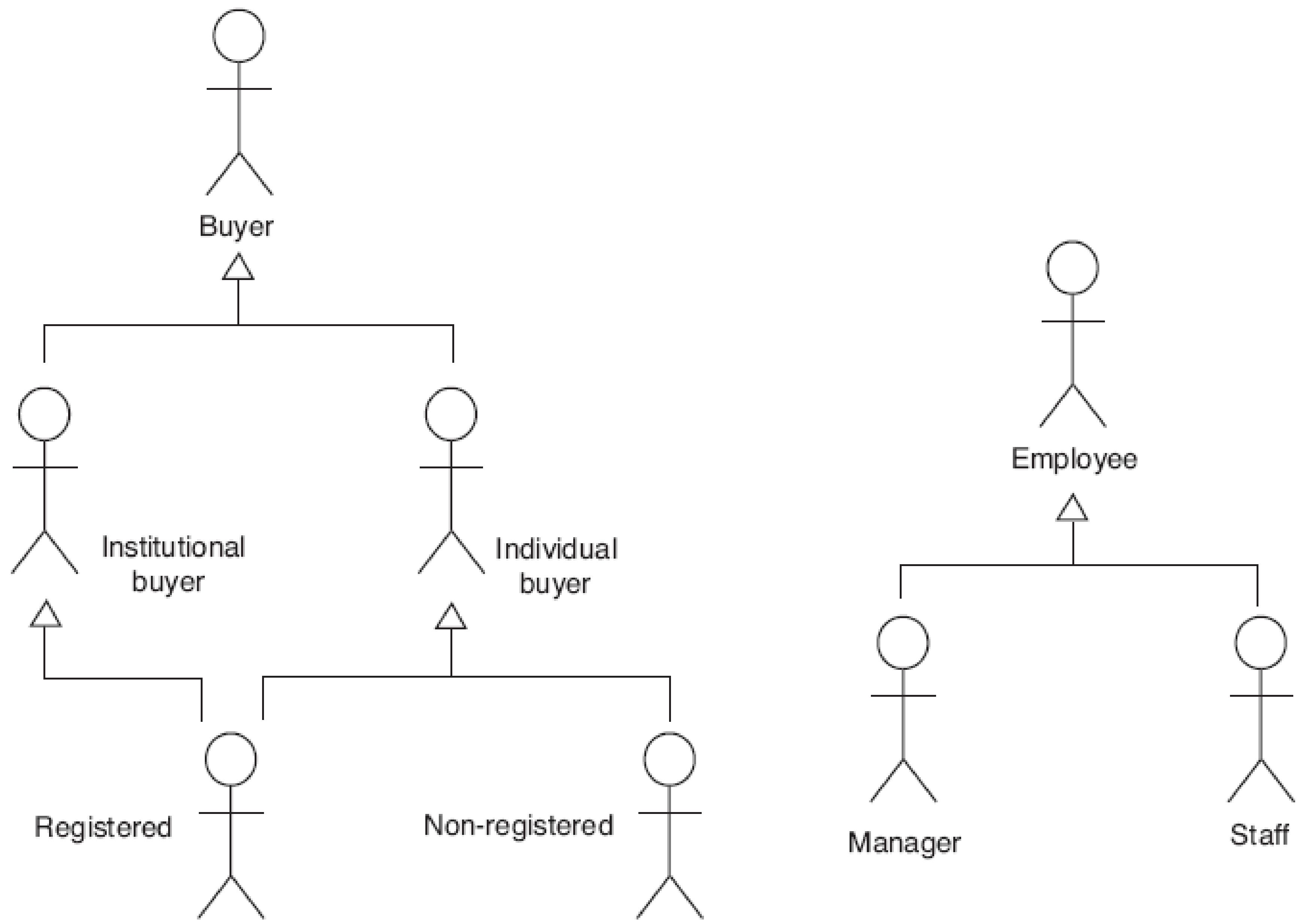

Figure 3.4 Generalization relationships among actors in the OPS system

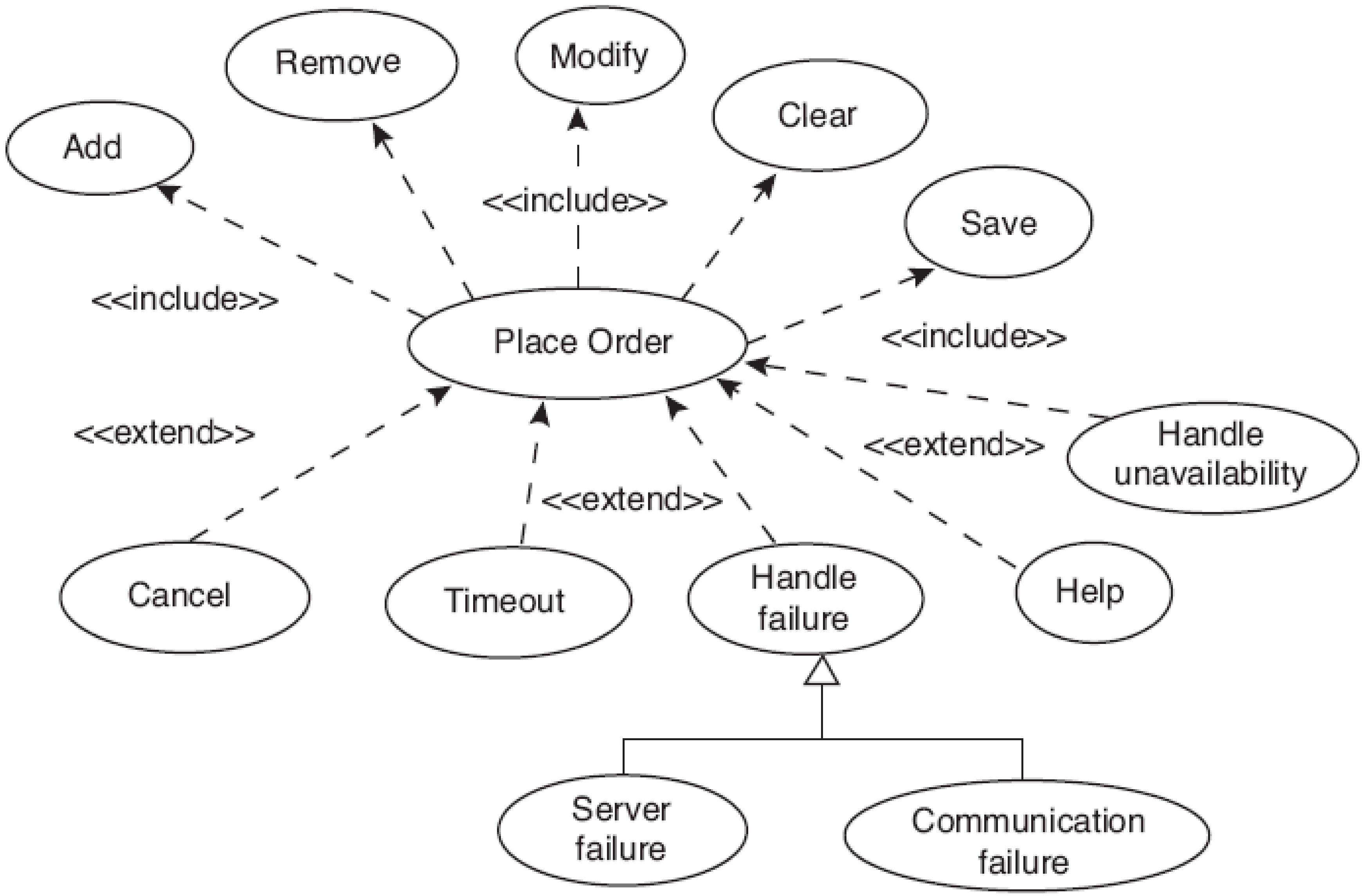

Figure 3.6 Use case Place Order with its include and extend relationships

Example

- Consider the following English description of the Online Publisher Storefront (OPS). The OPS is a software system developed primarily to allow buyers to place book orders online. Buyers can be individual buyers or institutional buyers, for example, bookstore managers. Individual buyers have to supply payment information when they place their orders. Also, if registered, and after making purchases for more than $1000, they have special discounts and limited credit facility. However, institutional buyers must be pre-registered and must have a good credit standing. Buyers credit facilities and discounts are substantial.

- Once an order is placed, the credit of the buyer is checked by the finance department and the order becomes active if the credit is in good standing. The order is then sent to the order fulfillment department for processing. Once the order is placed, the buyer has 24 hours to cancel or modify the order. The order fulfillment department is informed accordingly. When the order is placed, modified, or cancelled, the buyer is notified by an email that includes the order number in question. Once an order is fulfilled, it becomes a sale. Prior to placing an order, buyers normally browse the OPS catalogues to search for and select books and then place them in their electronic basket.

The content of the electronic basket can be modified at any time prior to the user placing the order. Before confirming an item in the basket, the system checks for its availability in the warehouse. Buyers can enquire about their orders using the order number. Additionally, users can e-mail a complaint or complete a form to complain about the quality of the service or dissatisfaction with the books they have received. The complaints are saved and dealt with by the OPS management staff . Managers and staff of OPS are registered users with two sets of privileges. Managers can modify the book catalogues and create, delete, and modify the staff records. Managers can also generate monthly reports for online sales. Registered users can enquire about their orders, view their purchasing history, and change their personal information.

Nouns and noun phrases

- OPS
- software system
- buyer
- book order
- individual buyers
- institutional buyer
- bookstore manager
- payment information
- purchase
- special discount
- limited credit facility
- good credit standing
- finance department
- order fulfillment department
- email, order number
- catalogue
- book
- electronic basket
- user
- item
- warehouse
- complaint
- form
- quality of the service
- management staff
- sets of privileges
- staff records
- monthly report
- purchasing history
- personal information

Table 3.2 Identification of actors in the OPS system

Nouns	Remarks	Actors
OPS Software system	Refer to the whole system	None
Buyer Individual buyer Institutional buyer	Linked by generalization relationship	Buyer Individual buyer Institutional buyer
Employee Bookstore manager Staff	Linked by generalization relationship	Employee Manager Staff
Warehouse	External system	Warehouse
Finance department	External system	Finance
Order fulfillment department	External system	Order fulfillment

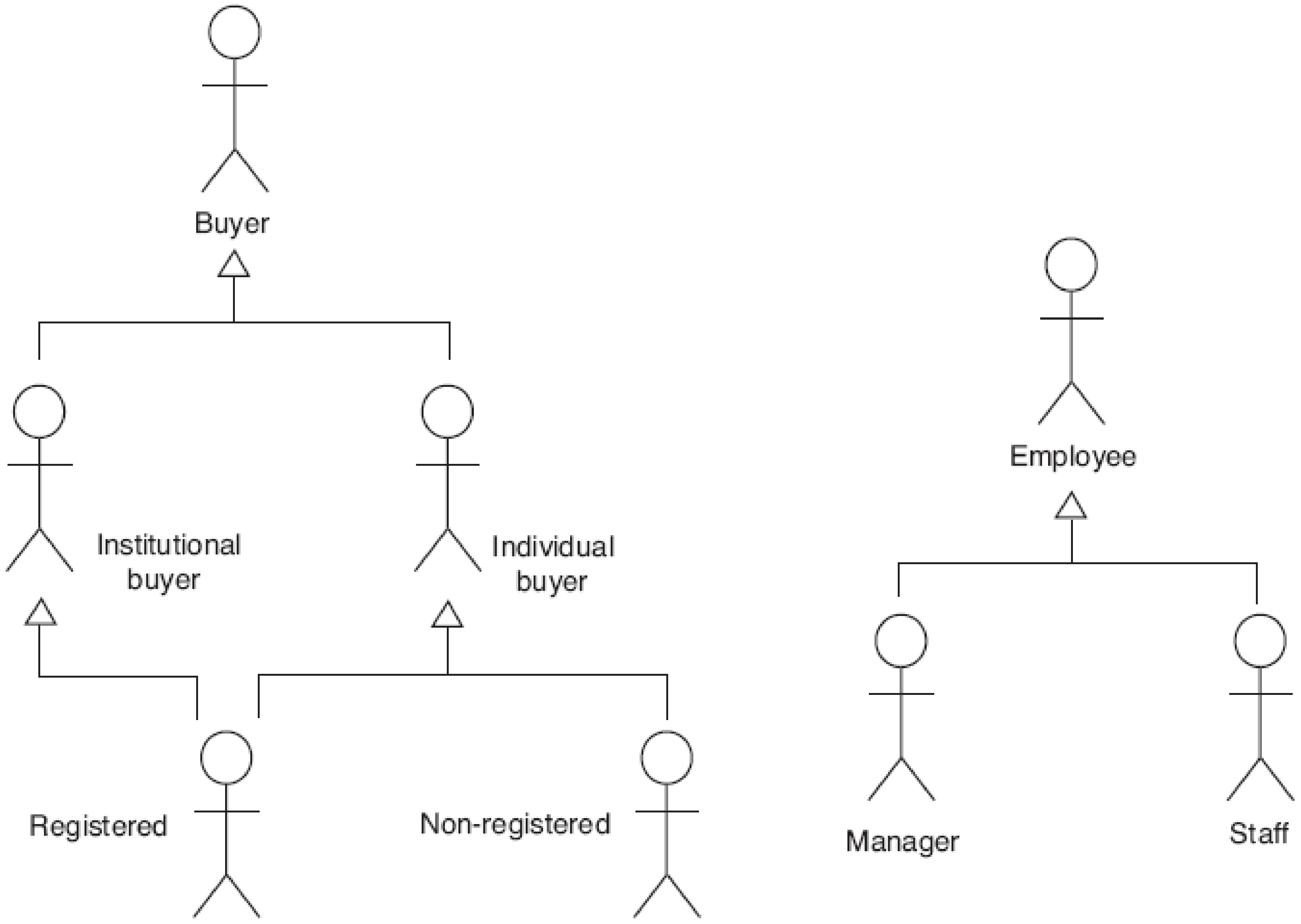

Figure 3.4 Generalization relationships among actors in the OPS system

Table 3.3 Identification of use cases in the OPS system

Use case	Triggering actor	Affected actors
Place order Modify order Cancel order Enquire about order	Buyer	Finance Warehouse
Enquire about order	Buyer	
Register	Buyer	
Browse catalogue	Buyer	Warehouse
E-mail buyer	System	Buyer
Check book availability	System	Warehouse
Provide payment information	System	Buyer
Check credit information	System	Finance
File complaint	Buyer	Manager
Check complaint	Manager	Buyer
Modify catalogue	Manager	Warehouse
Create, modify, or delete staff record	Manager	Staff
Generate monthly report	Manager	
View purchasing history	Registered buyer	
Change personal information	Registered buyer	

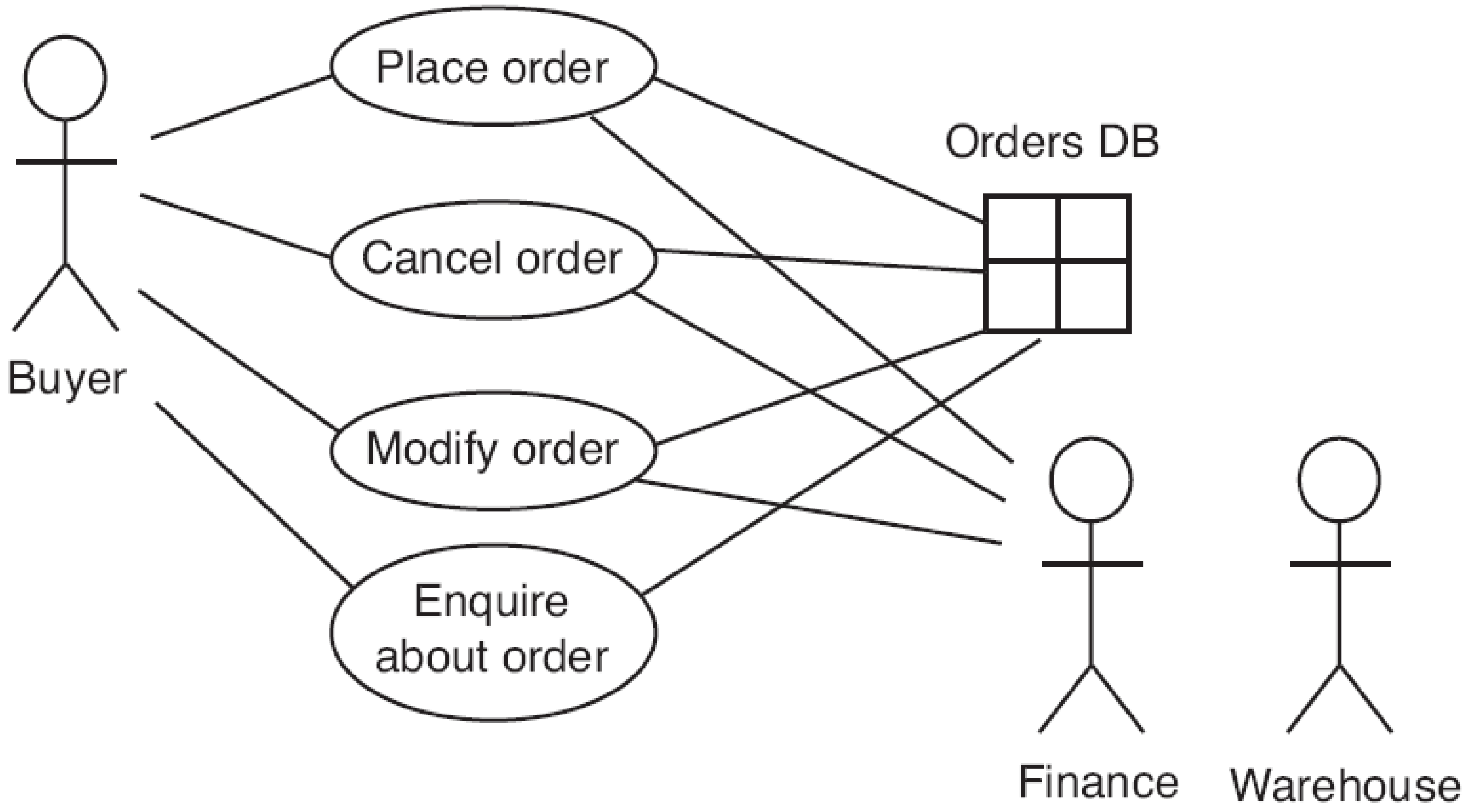

Figure 3.5 UCD including order-related use cases

Use case development

- Template for describing use cases
- Reducing the complexity of use cases
  - Use <<include>> relationships
  - Modular use cases

Table 3.4 Template for the description of a use case

Identification	
Name	
Created by	
Date created	
Updated by	
Date of update	
Change history	
Stakeholders	
Actors involved Primary Secondary or triggering Affected	
Brief description	
Assumption(s)	
Precondition(s)	
Triggering event(s)	
Postcondition(s)	
Priority	
Frequency of use	
Normal flow of events (scenarios)	
Alternative or abnormal flow of events	
Used use cases (include)	
Interrupting use cases (extend)	
Non-functional requirements and constraints	
Attached sequence diagram(s)	
Attached activity diagram(s)	

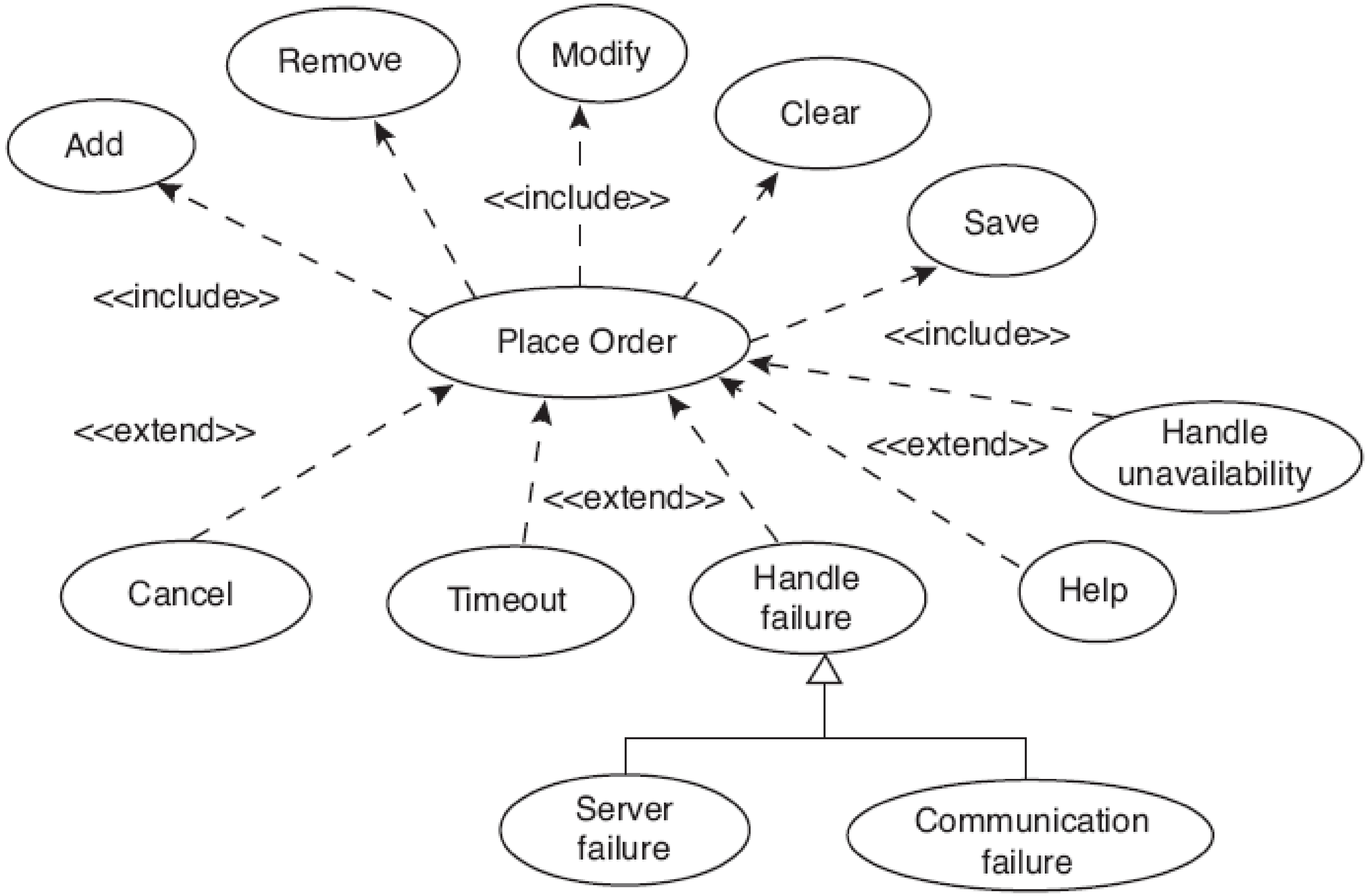

Figure 3.6 Use case Place Order with its include and extend relationships

Table 3.5 Place Order use case

Identification	UC x
Name	Place Order
Created by Date created Updated by Date of update	J. Smith, January 5, 2006
Actors involved	Triggering actor: Buyer Secondary (affected) actors: Finance and warehouse
Brief description	This use case is started by the buyer to construct electronic basket with books selected from the catalogue
Assumptions	Catalogue includes available books
Preconditions	Buyer could be a registered or unregistered individual user Buyer could be a registered institutional user Registered buyer has already logged on successfully
Postconditions	Electronic basket is created and closed prior to placing the order
Priority	High
Frequency of use	High (one hundred orders per day during the first year)
Flow of events (or steps)	1. Browse book catalogue 2. Select books 2.1. Select a book and quantity 2.2. Confirm availability 2.3. Compute current total cost 2.4. Repeat 2.1 through 2.3 until desired books are purchased 3. Confirm order 4. Provide payment information and get confirmation of payment

Alternative flows	Use case can be interrupted at any point by the use cases E1 to E7. Use case is discontinued when interrupted by the use cases: E1 to E4. However, the use case continues after being interrupted by use cases E5 to E7.
Used use cases (include)	Add book to basket Remove book from basket Modify order Clear order Save order Send order Provide payment information Check credit information Check book availability
Interrupting or extending use cases (extend)	E1: Handle timeout E2: Handle communications failure E3: Handle server failure E4: Handle cancel E5: Handle invalid payment information E6: Handle book unavailability E7: Help place order
Non-functional requirements	See Section 3.5
Remarks and issues	

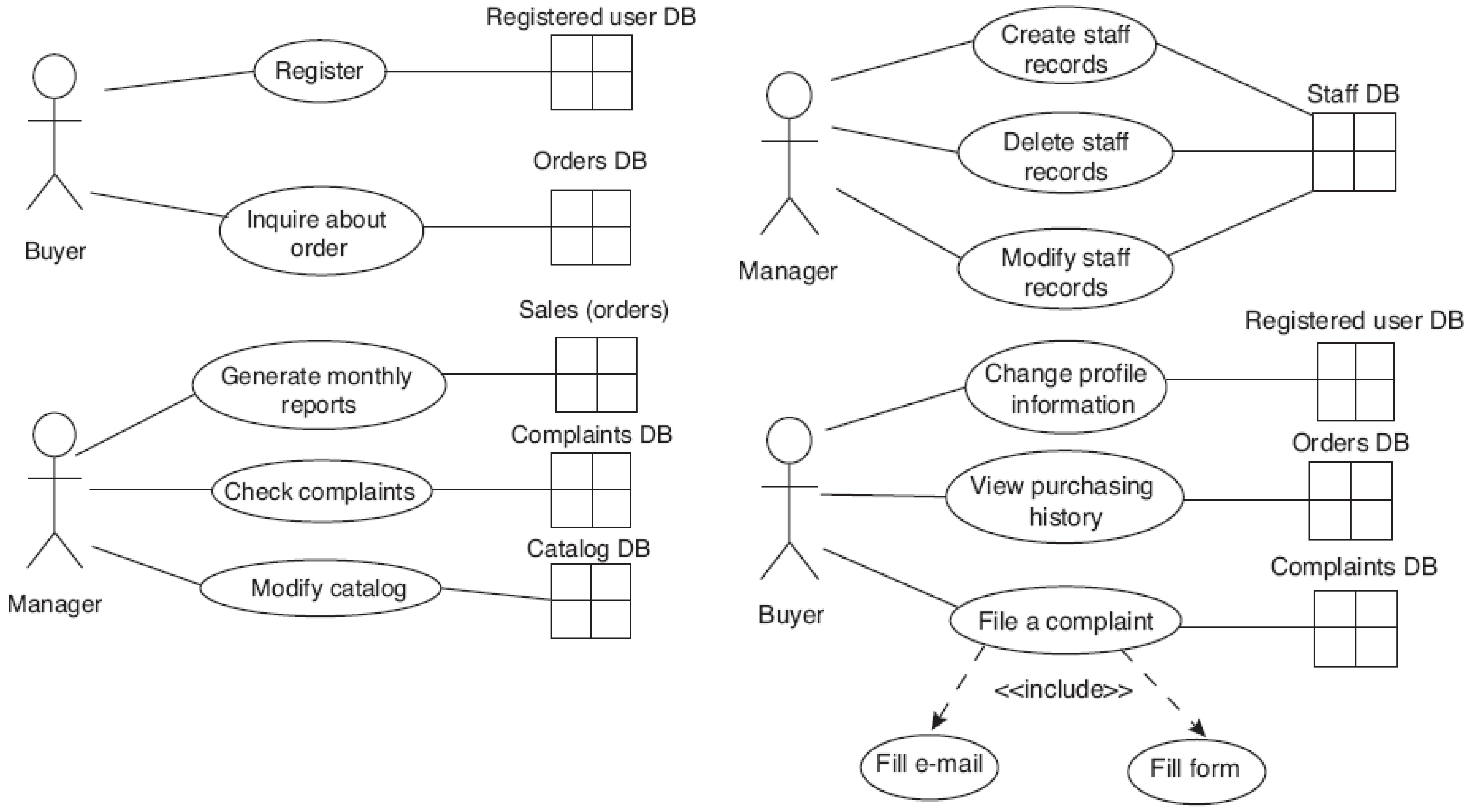

Figure 3.7 Additional use cases in the OPS system