Chapter 8 UNDERSTANDING REQUIREMENTS

8.1 REQUIREMENTS ENGINEERING

The broad spectrum of tasks and techniques that lead to an understanding of requirements is called requirements engineering. From a software process perspective,requirements engineering is a major software engineering action that begins during the communication activity and continues into the modeling activity. It must be adapted to the needs of the process, the project, the product, and the people doing the work.

Requirements engineering encompasses seven distinct tasks: inception, elicitation,elaboration, negotiation, specification, validation, and management. It is important to note that some of these tasks occur in parallel and all are adapted to the needs of the project.

Chapter 8 UNDERSTANDING REQUIREMENTS

1. **Inception:**

At project inception, you establish a basic understanding of the problem,the people who want a solution, the nature of the solution that is desired, and the effectiveness of preliminary communication and collaboration between the other stakeholders and the software team.

2. **Elicitation:**

An important part of elicitation is to establish business goals. Your job is to engage stakeholders and to encourage them to share their goals honestly. Once the goals have been captured, a prioritization mechanism should be established,and a design rationale for a potential architecture (that meets stakeholder goals，如，软件系统架构2.jpg) can be created. Christel and Kang identify a number of problems that are encountered as elicitation occurs.

Chapter 8 UNDERSTANDING REQUIREMENTS

Christel and Kang identify a number of problems that are encountered as elicitation occurs:

Problems of scope,

Problems of understanding,

Problems of volatility.

3. **Elaboration:**

The information obtained from the customer during inception and elicitation is expanded and refined during elaboration. This task focuses on developing a refined requirements model (Chapters 9 through 11) that identifies various aspects of software function, behavior, and information.

Elaboration is driven by the creation and refinement of user scenarios that describe how the end user (and other actors) will

interact with the system. Each user scenario is parsed to extract analysis classes—business domain entities that are visible to the end user. The attributes of each analysis class are defined, and the services that are required by each class are identified. The relationships and collaboration between classes are identified, and a variety of supplementary diagrams are produced.

4. **Negotiation:**

You have to reconcile these conflicts through a process of negotiation. Customers, users, and other stakeholders are asked to rank requirements and then discuss conflicts in priority. Using an iterative approach that prioritizes requirements,assesses their cost and risk, and addresses internal conflicts, requirements are eliminated, combined, and/or modified so that each party achieves some measure of satisfaction.

Chapter 8 UNDERSTANDING REQUIREMENTS

5. **Specification:**

In the context of computer-based systems (and software), the term specification means different things to different people. A specifi cation can be a written document, a set of graphical models, a formal mathematical model,a collection of usage scenarios, a prototype, or any combination of these.

6. **Validation:**

The work products produced as a consequence of requirements engineering are assessed for quality during a validation step. Requirements validation examines the specification to ensure that all software requirements have been stated unambiguously; that inconsistencies, omissions, and errors have been detected and corrected; and that the work products conform to the standards established for the process, the project, and the product.

Chapter 8 UNDERSTANDING REQUIREMENTS

The primary requirements validation mechanism is the technical review.The review team that validates requirements includes software engineers, customers, users, and other stakeholders who examine the specification looking 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 requirements, or unrealistic (unachievable) requirements.

7. **Requirements management:**

Requirements for computer-based systems change,and the desire to change requirements persists throughout the life of the system. Requirements management is a set of activities that help the project team identify,control, and track requirements and changes to requirements at any time as the project proceeds. Many of these

Chapter 8 UNDERSTANDING REQUIREMENTS

activities are identical to the software configuration management (SCM) techniques.

8.2 ESTABLISHING THE GROUNDWORK(Inception)

8.2.1 Identifying Stakeholders

8.2.2 Recognizing Multiple Viewpoints

8.2.3 Working toward Collaboration

8.2.4 Asking the First Questions

8.2.5 Nonfunctional Requirements

A nonfunctional requirement (NFR) can be described as a quality attribute, a performance attribute, a security attribute...,(In Section 8.3.2),Quality function deployment (QFD) attempts to translate unspoken customer needs or goals into system requirements. Nonfunctional requirements are often listed separately in a software requirements specification.

Chapter 8 UNDERSTANDING REQUIREMENTS

8.2.6 Traceability(read it by yourself)

Traceability is a software engineering term that refers to documented links between software engineering work products (e.g., requirements and test cases).A traceability matrix allows a requirements engineer to represent the relationship between requirements and other software engineering work products.

The traceability matrices can support a variety of engineering development activities. They can provide continuity for developers as a project moves from one project phase to another, regardless of the process model being used. Traceability matrices often can be used to ensure the engineering work products have taken all requirements into account.

Chapter 8 UNDERSTANDING REQUIREMENTS

8.3 ELICITING REQUIREMENTS(elicitation)

8.3.1 Collaborative Requirements Gathering

A one- or two-page "product request" is generated during inception. A meeting place, time, and date are selected; a facilitator is chosen; and attendees from the software team and other stakeholder organizations are invited to participate. The product request is distributed to all attendees before the meeting date.

Before the meeting, each attendee is asked to make:

a list of objects,

a list of services (processes or functions),

a list of constraints (e.g., cost, size,business rules),

performance and other nonfunction requirements(e.g., speed, accuracy).

Chapter 8 UNDERSTANDING REQUIREMENTS

In many cases, an object or service described on a list will require further explanation to develop mini-specifications. For example:

Our research indicates that the market for home management systems is growing at a rate of 40 percent per year. The first SafeHome function we bring to market should be the home security function. Most people are familiar with "alarm systems" so this would be an easy sell.

The home security function would protect against and/or recognize a variety of undesirable "situations" such as illegal entry, fire, flooding, carbon monoxide levels, and others. It'll use our wireless sensors to detect each situation, can be programmed by the homeowner, and will automatically telephone a monitoring agency when a situation is detected.

Chapter 8 UNDERSTANDING REQUIREMENTS

8.3.2 Quality Function Deployment(read it by yourself)

QFD is a quality management technique that translates the needs of the customer into technical requirements for software. QFD "concentrates on maximizing customer satisfaction from the software engineering process". To accomplish this, QFD emphasizes an understanding of what is valuable to the customer and then deploys these values throughout the engineering process.

Within the context of QFD , normal requirements identify the objectives and goals that are stated for a product or system during meetings with the customer.If these requirements are present, the customer is satisfied. Expected requirements are implicit to the product or system and may be so fundamental that the customer does not explicitly state them. Their absence will be a cause for significant dissatisfaction. Exciting requirements go beyond the customer's expectations and prove to be very satisfying when present.

Chapter 8 UNDERSTANDING REQUIREMENTS

Although QFD concepts can be applied across the entire software process; specific QFD techniques are applicable to the requirements elicitation activity. QFD uses customer interviews and observation, surveys, and examination of historical data (e.g., problem reports) as raw data for the requirements gathering activity. These data are then translated into a table of requirements—called the customer voice table —that is reviewed with the customer and other stakeholders. A variety of diagrams, matrices, and evaluation methods are then used to extract expected requirements and to attempt to derive exciting requirements.

Chapter 8 UNDERSTANDING REQUIREMENTS

8.3.3 Usage Scenarios(Developing a Preliminary User Scenario(use case diagram))

As requirements are gathered, it is difficult to move into more technical software engineering activities until you understand how these functions and features will be used by different classes of end users. To accomplish this, developers and users can create a set of scenarios that identify a thread of usage for the system to be constructed. Use cases are discussed in greater detail in Section 8.4.

Chapter 8 UNDERSTANDING REQUIREMENTS

8.3.4 Elicitation Work Products(requirement specification)

The work products produced as a consequence of requirements elicitation will vary depending on the size of the system or product to be built. For most systems,the work products include: (1) a statement of need and feasibility, (2) a bounded statement of scope for the system or product, (3) a list of customers, users, and other stakeholders who participated in requirements elicitation, (4) a description of the system's technical environment, (5) a list of requirements (preferably organized by function) and the domain constraints that applies to each, (6) a set of usage scenarios that provide insight into the use of the system or product under different operating conditions, and (7) any prototypes developed to better define requirements. Each of these work products is reviewed by all people who have participated in requirements elicitation.

Chapter 8 UNDERSTANDING REQUIREMENTS

8.3.5 Agile Requirements Elicitation(read it by yourself)

Within the context of an agile process, requirements are elicited by asking all stakeholders to create user stories . Each user story describes a simple system requirement written from the user's perspective. User stories can be written on small note cards, making it easy for developers to select and manage a subset of requirements to implement for the next product increment.

Although the agile approach to requirements elicitation is attractive for many software teams, critics argue that a consideration of overall business goals and nonfunctional requirements is often lacking.

8.3.6 Service-Oriented Methods(read it by yourself)

Requirements elicitation in service-oriented development focuses on the definition of services to be rendered by an application.

Chapter 8 UNDERSTANDING REQUIREMENTS

8.4 DEVELOPING USE CASES(how to develop use case diagram)

A use case tells a stylized story about how an end user (playing one of a number of possible roles) interacts with the system under a specific set of circumstances.

- ***actor,primary actor,secondary actor***
- ***user(end user)***

Once actors have been identified, use cases can be developed. Jacobson suggests a number of questions that should be answered by a use case:

- **Who is the primary actor, the secondary actor(s)?**
- **What are the actor's goals?**
- **What preconditions should exist before the story begins?**

Chapter 8 UNDERSTANDING REQUIREMENTS

- What main tasks or functions are performed by the actor?
- What exceptions 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?

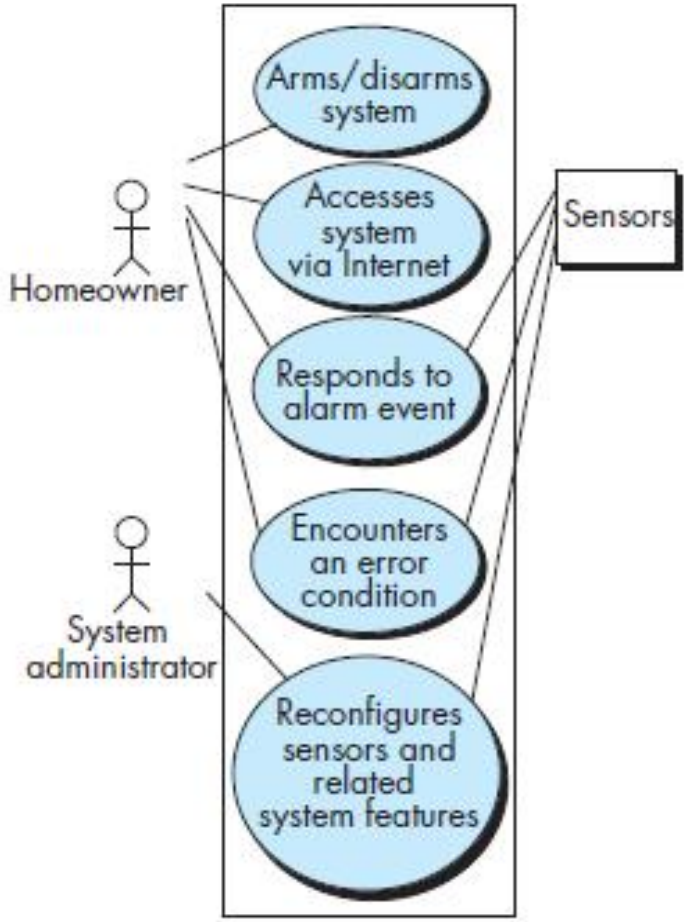
Give an example: 8章附件1-图书馆借阅管理系统.doc

Exercise:Pls reanalyze the use case diagram for SafeHome home security function.

Chapter 8 UNDERSTANDING REQUIREMENTS

FIGURE 8.2

UML use case diagram for *SafeHome* home security function



Chapter 8 UNDERSTANDING REQUIREMENTS

8.5 BUILDING THE ANALYSIS MODEL(elaboration)

The intent of the analysis model is to provide a description of the required informational, functional, and behavioral domains for a computer-based system.

8.5.1 Elements of the Analysis Model

Different modes of representation force you to consider requirements from different viewpoints—an approach that has a higher probability of uncovering omissions, inconsistencies, and ambiguity. A set of generic elements is common to most analysis models.

Scenario-based elements:

use case diagram

function elements

activity diagram(function model, depicts the business logic for each use case).

Chapter 8 UNDERSTANDING REQUIREMENTS

Class-based elements（information or data elements）:

class diagram(information model).

Behavioral elements:

state diagram and sequence diagram.

① **The behavior of a computer-based system can have a profound effect on the design that is chosen and the implementation approach that is applied. Therefore, the requirements model must provide modeling elements that depict behavior. The state diagram is one method for representing the behavior of a system by depicting its states and the events that cause the system to change state. A state is any observable mode of behavior. In addition, the state diagram indicates what actions (e.g., process activation) are taken as a consequence of a particular event.**

Chapter 8 UNDERSTANDING REQUIREMENTS

② In addition to behavioral representations of the system as a whole, the behavior of individual classes can also be modeled and represent the call relationships among different methods （StudentGrade class-initgra(),addgra(),delegra(),inqgra(),modgra(),sendgra()）.

Sequence diagram is one type of behavoiral model.

8.5.2 Analysis Patterns(read it by yourself)

Anyone who has done requirements engineering on more than a few software projects begins to notice that certain problems reoccur across all projects within a specific application domain.These analysis patterns suggest solutions (e.g., a class, a function, a behavior) within the application domain that can be reused when modeling many applications.

8.5.3~8.5.4(read it by yourself)

Chapter 8 UNDERSTANDING REQUIREMENTS

8.6 NEGOTIATING REQUIREMENTS(negotiation)

Boehm defines a set of negotiation activities:

① **Identification of the system or subsystem's key stakeholders.**

② **Determination of the stakeholders' "win conditions."**

③ **Negotiation of the stakeholders' win conditions to reconcile them into a set of win-win conditions for all concerned (including the software team).**

8.7 REQUIREMENTS MONITORING(read it by yourself)

Today, incremental development is commonplace. This means that use cases evolve, new test cases are developed for each new software increment, and continuous integration of source code occurs throughout a project. Requirements monitoring can be extremely useful when incremental development is used. It

Chapter 8 UNDERSTANDING REQUIREMENTS

encompasses five tasks: (1) distributed debugging uncovers errors and determines their cause, (2) run-time verification determines whether software matches its specification, (3) run-time validation assesses whether the evolving software meets user goals, (4) business activity monitoring evaluates whether a system satisfies business goals, and (5) evolution and codesign provides information to stakeholders as the system evolves.

8.8 VALIDATING REQUIREMENTS(validation)

As each element of the requirements model is created,A review of the requirements model addresses the following questions:

- **Is each requirement consistent with the overall objectives for the system or 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?**

Chapter 8 UNDERSTANDING REQUIREMENTS

- 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?
- 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?

Chapter 8 UNDERSTANDING REQUIREMENTS

- **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?**