4: Understanding Requirements

Requirements Engineering:

- Inception: Understand the problem and the people who want a solution.
- Elicitation: Elicit requirements and business goals form from stakeholders.
- Elaboration: Focuses on developing a refined(세련된) requirements model that I identifies aspects of software function, behaviour, and information.
- Negotiation: Agree the system upon developers and customers
- Specification: Written documents, graphical models, mathematical models.
- Validation: Requirements engineering work products produced during requirements are quality products.
- Requirements management : Set of traceability activities to help the project team identify, control and track requirements and their changes.

Non-Functional Requirements: Quality attribute, performance attribute, security attribute, or general system constraint.

Establishing the Groundwork

- Identify stakeholders.
- Recognize multiple points of view
- Work toward collaboration
- Questions: who is behind the request for this work?, who will use this solution? .. etc

Collaborative requirements gathering

- Meetings are conducted and attended by both programmers and stakeholders.
- Rules for preparation and participation are established.
- Agenda is suggested that is formal enough to cover all important points.

Elicitation work products

- Statement of need and feasibility (실현가능성)
- Bounded statement of scope for the system or product.
- List of customers, users, and other stakeholders who participated in requirements elicitation
- Description of the system's technical environment
- List of requirements and the domain constraints that apply to each
- Set of usage scenarios that provide insight into the use of the system or product under different operating conditions.

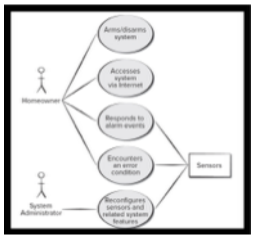
Use Case Definition

- 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.

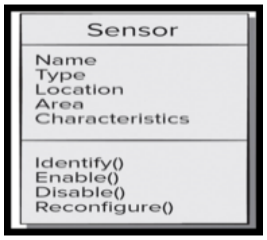
Analysis Model Elements

Analysis model: provides a description of the required informational, functional, and behavioural domains for a computer-based system,
Scenario-based elements: functional descriptions are expressed in the customers own words and user stories and as interactions of actor with the system expressed using UML use case diagrams.
Class-based elements: collections of attributes and behaviours implied by the user stories and expressed using UML class diagrams
Behavioural elements: may be expressed using UML state diagrams as inputs causing state changes.

UML use case diagram


Arms/disarms system
Accesses system via Internet
Homeowner
Responds to alarm events
Encounters an error condition
Sensors
System Administrator
Reconfigures sensors and related system features

UML Class Diagram



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 Cana be used to address the problem.
Forces and context: A description of external issues that can affect how the pattern is used and the external issue 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 behavioural 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: One 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;

Negotiating Requirements

- Negotiations strive(애쓰다) for a "win-win" result, stakeholders win by getting satisfied product and developers win by getting achievable deadlines.
- Handshaking is one-way to achieve "win-win"
  - Developers propose solutions to requirements, describe their impact and communicate their intentions to the customers.
  - Customer review the proposed solutions, focusing on missing features and seeking clarification of novel requirements.
  - Requirements are determined to be good enough if the customers accept the proposed solutions.

Requirements Monitoring

1. Distributed debugging - uncovers errors and determines their cause.
2. Run-time verification - determines whether software matches its specification
3. Run-time validation - assess whether the evolving software meets user goals.
4. Business activity monitoring - evaluates whether a system satisfies business goals.
5. Evolution and codesign - provides information to stakeholders as the system evolves.

Validating Requirements

Validate if the each requirement consistent with the overall objective for the system/product. Is each requirement testable, once implemented?
Have all requirements been specified at the proper level of abstraction?
Is each requirement bounded and unambiguous?
Have requirements patterns been used to simplify the requirements model.