

Chapter 7

Requirements Engineering

Moonzoo Kim
CS Division of EECS Dept.
KAIST

moonzoo@cs.kaist.ac.kr

<http://pswlab.kaist.ac.kr/courses/CS350-07>

Requirements Engineering-I

- **Inception**—ask a set of questions that establish ...
 - basic understanding of the problem (what)
 - the people who want a solution (who)
 - the nature of the solution that is desired, and
 - 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 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 (ambiguity)
 - Missing information (incomplete requirement)
 - Inconsistencies
 - a major problem when large products or systems are engineered)
 - Unrealistic (unachievable) requirements.
- **Requirements management**

Inception

- Identify stakeholders
 - “who else do you think I should talk to?”
- Recognize multiple points of view
- 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?

Eliciting Requirements

- meetings are conducted and attended by both software engineers and customers
- 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

Conducting a Requirements Gathering Meeting (pg188)

■ The scene:

- A meeting room. The first requirements gathering meeting is in progress.

■ The players:

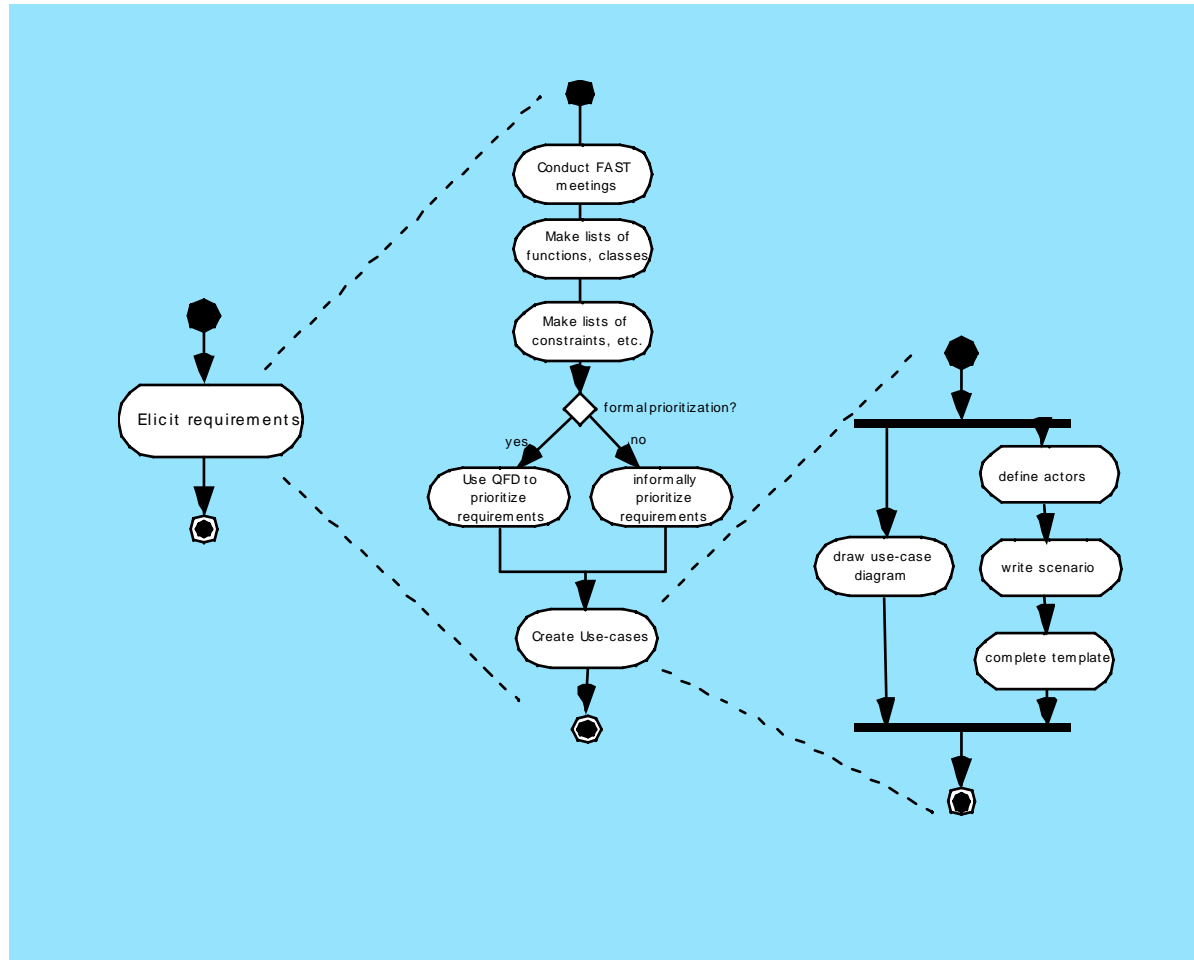
- **Jamie** Lazar, software team member;
- **Vinod** Raman, software team member;
- **Ed** Robbins, software team member;
- **Doug** Miller, software engineering manager;
- **three members of marketing**;
- a product engineering representative;
- a **facilitator**.

• The conversation:

- **Facilitator (pointing at white board)**: So that's the current list of objects and services for the home security function.
- **Marketing person**: That about covers it from our point of view.
- **Vinod**: Didn't someone mention that they wanted all *SafeHome* functionality to be accessible via the Internet? That would include the home security function, no?
- **Marketing person**: Yes, that's right ... we'll have to add that functionality and the appropriate objects.

- **Facilitator:** Does that also add some constraints?
- **Jamie:** It does, both technical and legal.
- **Production rep:** Meaning?
- **Jamie:** We better make sure an outsider can't hack into the system, disarm it, and rob the place or worse. Heavy liability on our part.
- **Doug:** Very true.
- **Marketing:** But we still need Internet connectivity . just be sure to stop an outsider from getting in.
- **Ed:** That's easier said than done and....
- **Facilitator (interrupting):** I don't want to debate this issue now. Let's note it as an action item and proceed. (Doug, serving as the recorder for the meeting, makes an appropriate note.)
- **Facilitator:** I have a feeling there's still more to consider here.
- (The group spends the next 45 minutes refining and ex-panding the details of the home security function.)

Eliciting Requirements



Quality Function Deployment

- **Function deployment** determines each function required of the system
- **Information deployment** identifies data objects and events
- **Task deployment** examines the behavior of the system
- **Value analysis** determines the relative priority of requirements during each of the three deployments
 - Value should be one that are perceived by the customer

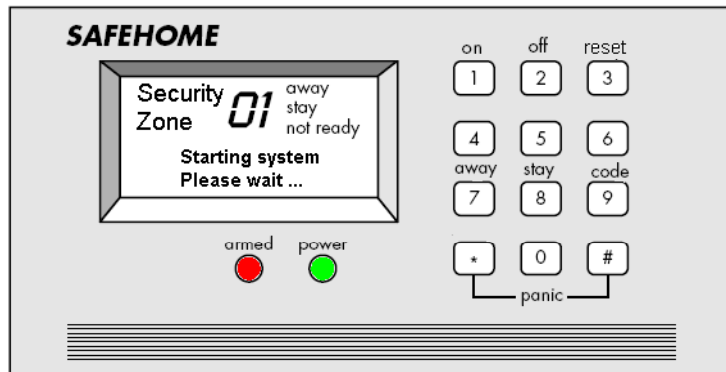
Elicitation Work Products

- 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.
- 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.

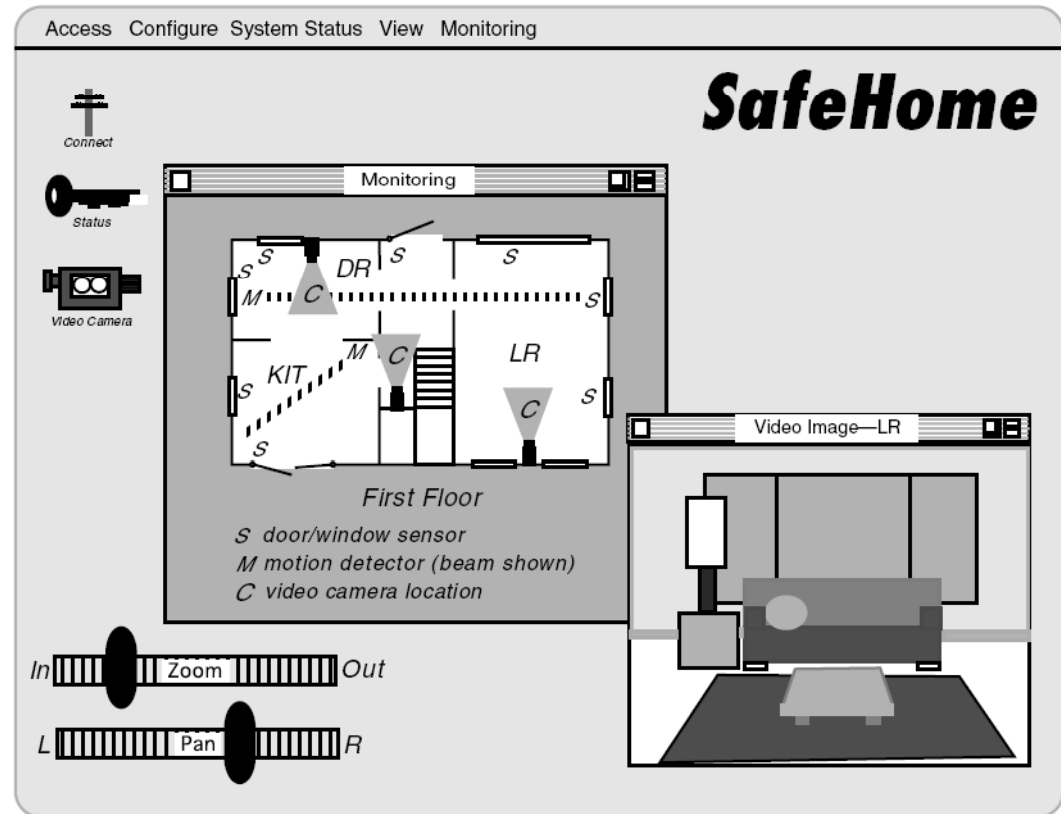
Use-Cases

- 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?

SafeHome Product



See SEPA 193,231pg for more details

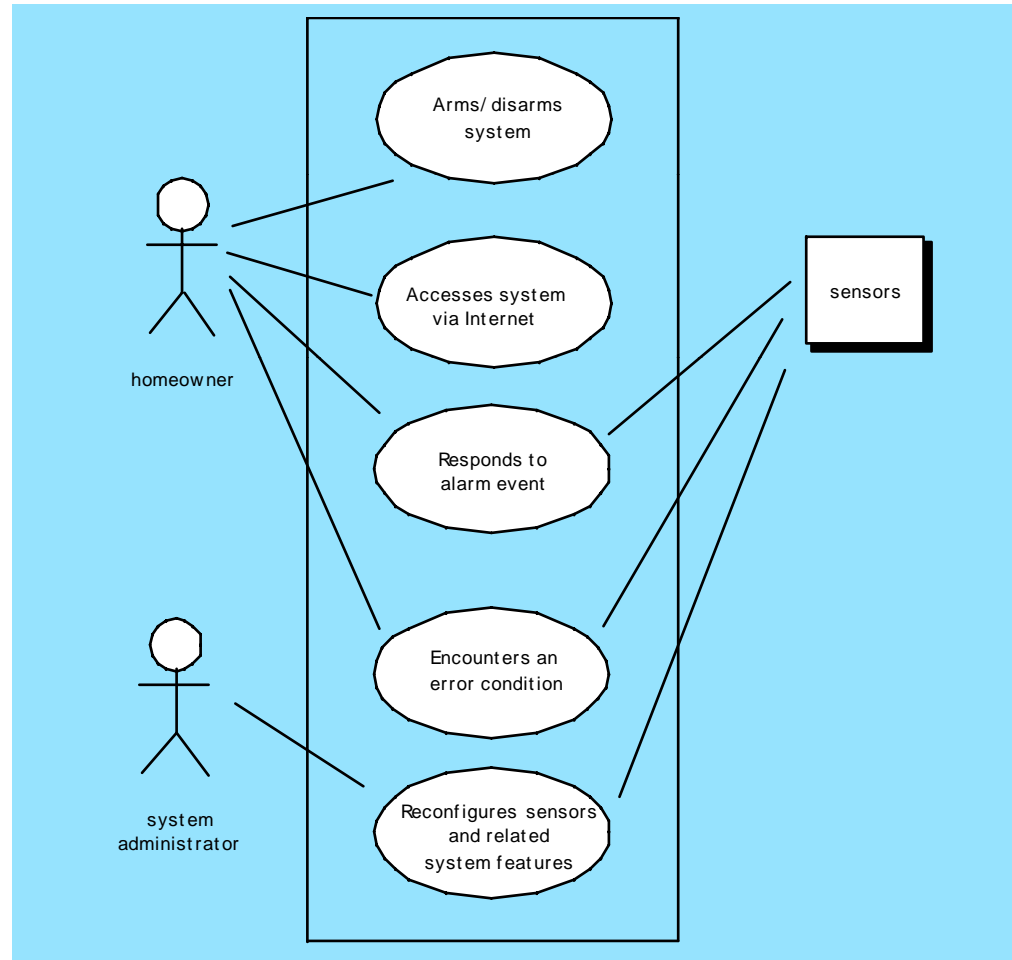


See 376 pg for more details

Example of Use Case for SafeHome

- **Use-case:** InitiateMonitoring
- **Primary actor:** Homeowner
- **Goal in context:** To set the system to monitor sensors when the homeowner leaves the house or remains inside
- **Preconditions:** System has been programmed for a password and to recognize various sensors
- **Trigger:** The homeowner decides to “set” the system, i.e., to turn on the alarm functions
- **Scenario:**
 1. Homeowner: observes control panel
 2. Homeowner: enters password
 3. Homeowner: selects “stay” or “away”
 4. Homeowner: observes red alarm light to indicate that SafeHome has been armed
- **Exceptions:**
 - 1a. Control panel is not ready: homeowner checks all sensors to determine which are open; closes them
 - 2a. Password is incorrect
- **Priority:** Essential, must be implemented
- **When available:** first increment
- **Frequency of use:** Many times per day
- **Channel to actor:** Via control panel interface
- **Secondary actors:** Support technician
- **Channels to secondary actors:** support technician: phone line
- **Open issues:**
 - Do we enforce time limit for password entering?

Use-Case Diagram

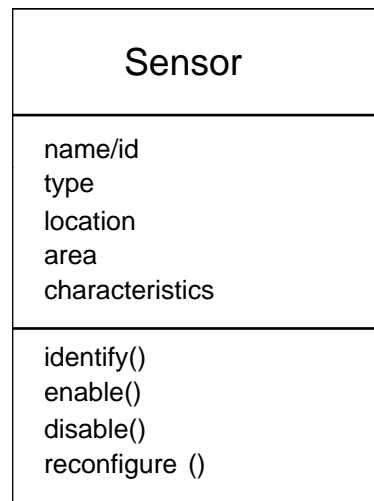


Building the Analysis Model

- 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 ...



State Diagram

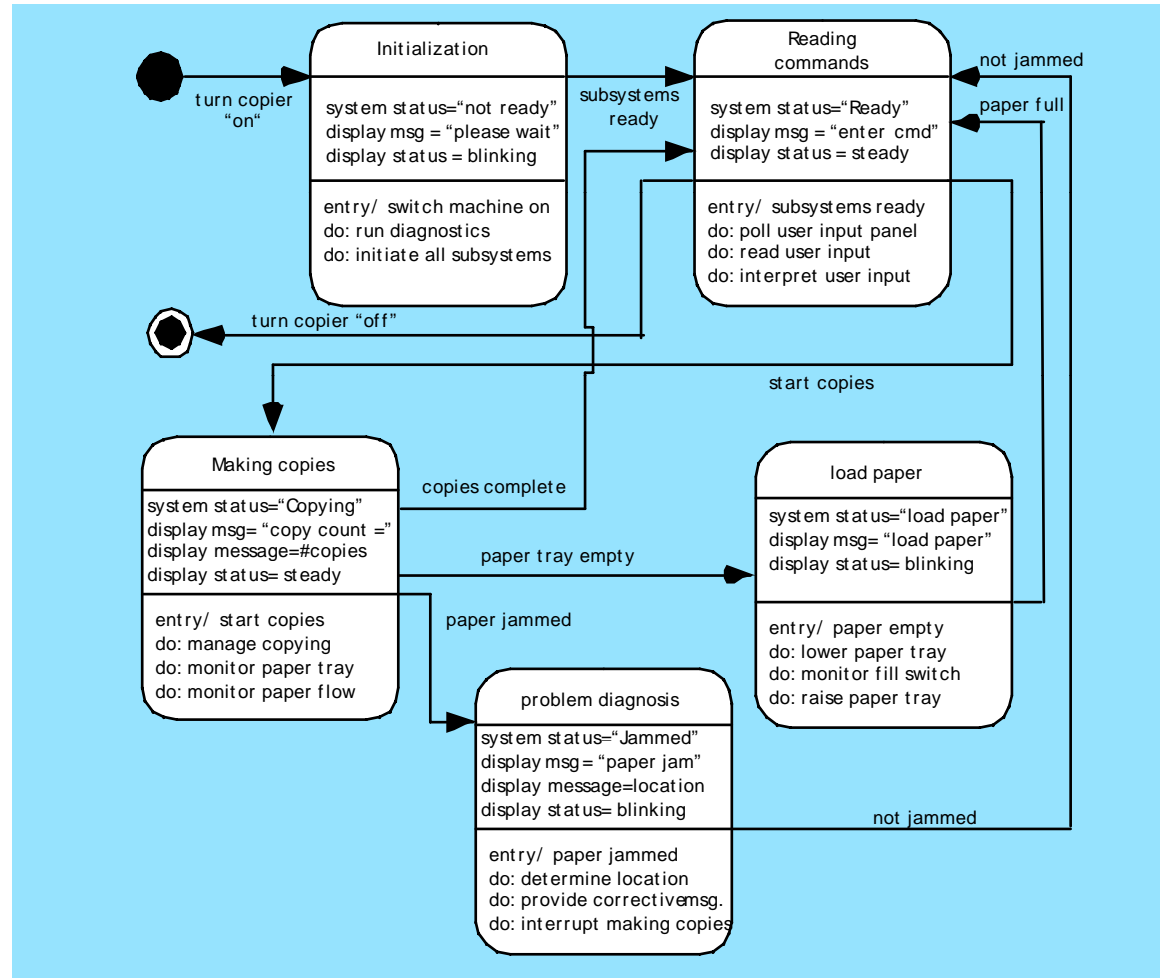


Figure 7.6 Preliminary UML state diagram for a photocopier

Negotiating Requirements

- **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”

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?

Validating Requirements-II

- 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.
- Has the requirements model been “partitioned” in a way that exposes progressively more detailed information about the system.