Understanding Requirements

Establishing the Groundwork

Dr. A. Joy Christy
AP II

- The broad spectrum of tasks and techniques that lead to an understanding of requirements is called requirements engineering

- Requirements engineering builds a bridge to design and construction

- begins from stakeholders where
  - business need is defined,
  - user scenarios are described,
  - functions and features are described,
  - project constraints are identified

- Requirements engineering encompasses seven distinct tasks:

✔ inception, elicitation, elaboration, negotiation, specification, validation, and management

- INCEPTION:
  - ask a set of questions that establish for
    - 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 other stakeholders and the software team.
- Elicitation:
  - Main part of elicitation is to establish the business goals
  - Task is to engage stakeholders to capture goals and prioritise them to have potential architectural style

- Elicitation:
  - Problems occurs at this task
    - Problems of scope : occurs when the boundary of the system is ill-defined
      - Unnecessary details may confuse overall system objective
    - Problems of understanding: occurs when the user is not sure about what is needed
    - Problems of Volatility: occurs when the requirements change over time
  - To overcome this, requirements gathering must be done in an organized manner.

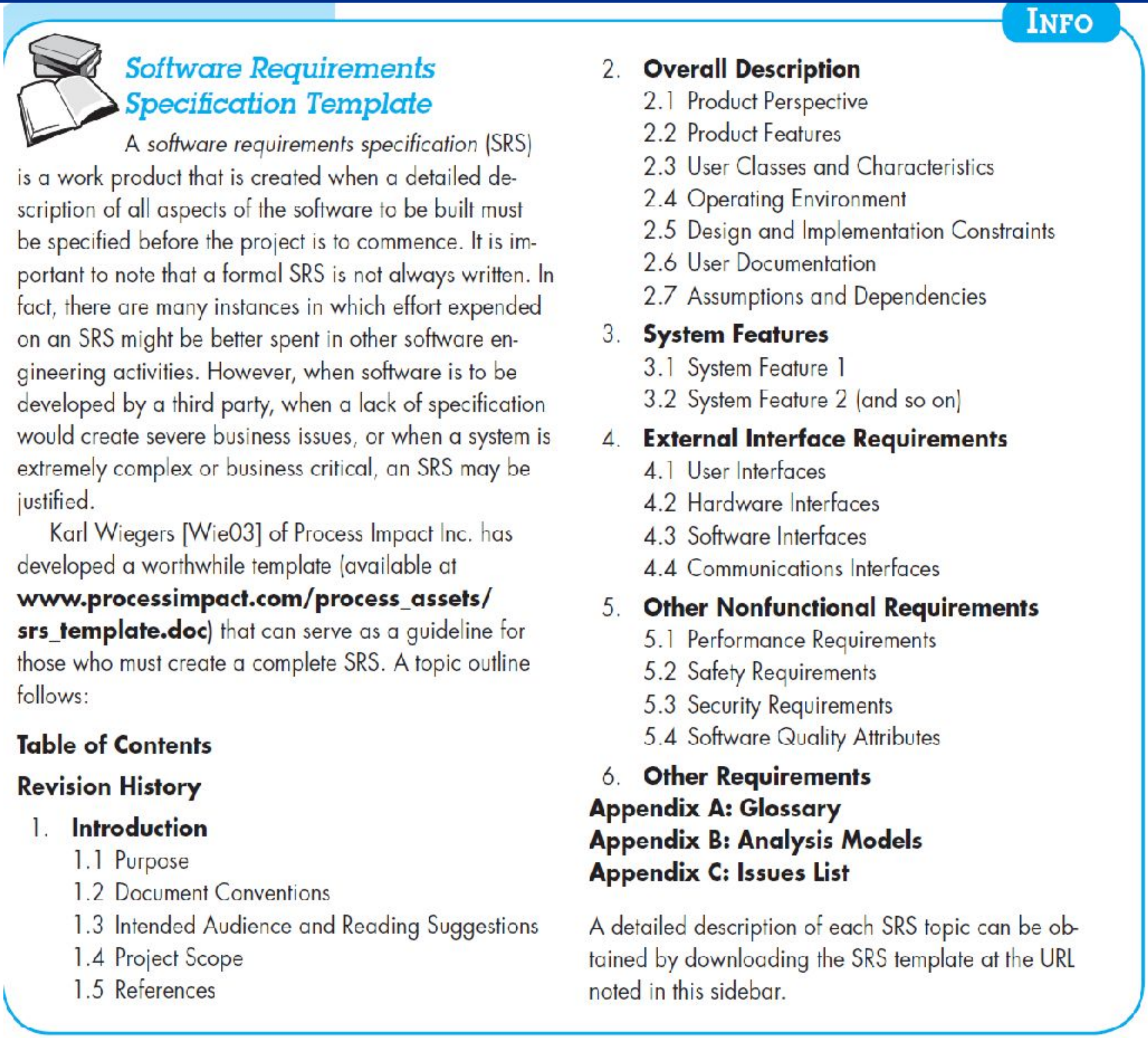
- Elaboration:
  - The information obtained from the customer during inception and elicitation is expanded and refined during elaboration
  - Refined requirements model is developed
  - creation and refinement of user scenarios that describe how the end user (and other actors) will interact with the system has been done
  - Deliverables would be several diagrams

- Negotiation:
  - agree on a deliverable system that is realistic for developers and customers

- Specification:
  - written document, a set of graphical models, a formal mathematical model, a collection of usage scenarios, a prototype, or any combination of these
  - *Template* should be developed to document the specification
  - *For large system :* written document and combination of natural language description and graphical model

- 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 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



INFO

Software Requirements Specification Template

A *software requirements specification* (SRS) is a work product that is created when a detailed description of all aspects of the software to be built must be specified before the project is to commence. It is important to note that a formal SRS is not always written. In fact, there are many instances in which effort expended on an SRS might be better spent in other software engineering activities. However, when software is to be developed by a third party, when a lack of specification would create severe business issues, or when a system is extremely complex or business critical, an SRS may be justified.

Karl Wiegers [Wie03] of Process Impact Inc. has developed a worthwhile template (available at **www.processimpact.com/process_assets/srs_template.doc**) that can serve as a guideline for those who must create a complete SRS. A topic outline follows:

Table of Contents

Revision History

1. **Introduction**
   - 1.1 Purpose
   - 1.2 Document Conventions
   - 1.3 Intended Audience and Reading Suggestions
   - 1.4 Project Scope
   - 1.5 References
2. **Overall Description**
   - 2.1 Product Perspective
   - 2.2 Product Features
   - 2.3 User Classes and Characteristics
   - 2.4 Operating Environment
   - 2.5 Design and Implementation Constraints
   - 2.6 User Documentation
   - 2.7 Assumptions and Dependencies
3. **System Features**
   - 3.1 System Feature 1
   - 3.2 System Feature 2 (and so on)
4. **External Interface Requirements**
   - 4.1 User Interfaces
   - 4.2 Hardware Interfaces
   - 4.3 Software Interfaces
   - 4.4 Communications Interfaces
5. **Other Nonfunctional Requirements**
   - 5.1 Performance Requirements
   - 5.2 Safety Requirements
   - 5.3 Security Requirements
   - 5.4 Software Quality Attributes
6. **Other Requirements**

Appendix A: Glossary
Appendix B: Analysis Models
Appendix C: Issues List

A detailed description of each SRS topic can be obtained by downloading the SRS template at the URL noted in this sidebar.

Inception

Identifying Stakeholders

- anyone who benefits in a direct or indirect way from the system
    - business operations managers,
    - product managers
    - marketing people
    - internal and external customers
    - end users, consultants
    - Product engineers
    - software engineers
    - support and maintenance engineers and others
- create a list of people who will contribute input as requirements are elicited

Recognizing Multiple Viewpoints

- the marketing group - easy to sell
- Business managers - within budget & manage market windows
- End users -easy to learn and use
- Software engineers – create marketable functions and infrastructure
- Support engineers - maintainability of the software

Working toward Collaboration

- The job of a requirements engineer is to identify areas of
    - commonality
    - conflict or inconsistency (can be solved by voting)
- business manager or a senior technologist may make the final decision

Asking the First Questions

- help to “break the ice” and initiate the communication
- Questions should be “context free”
- First set of questions focuses on project goals and benefits
    - 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?
- This helps identify all stakeholders who will have interest in the software to be built

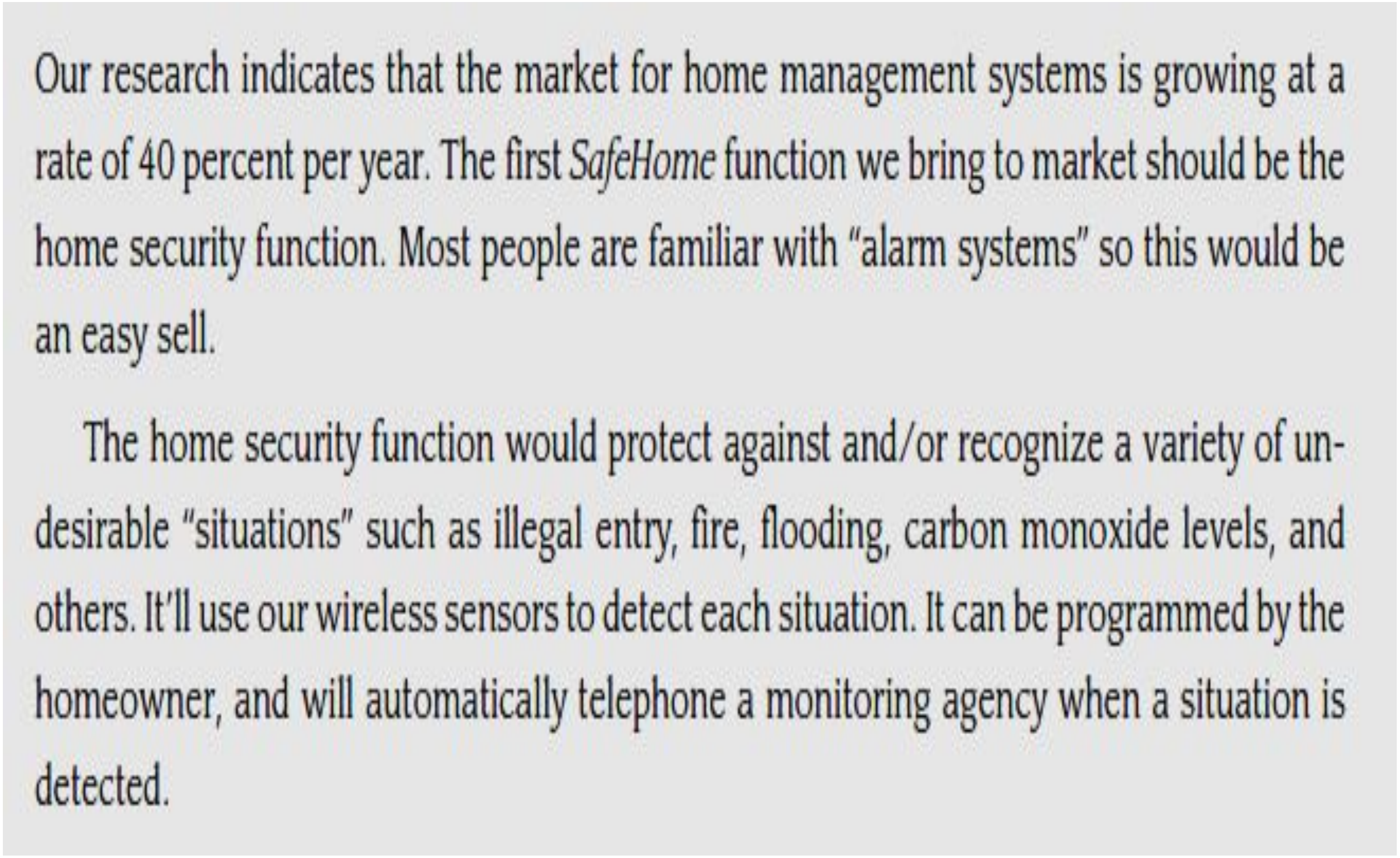
Asking the First Questions

- next set of questions enables you to gain a better understanding of the problem
  - How would you characterize “good” output?
  - What problem(s) will this solution address?
  - Can you show me (or describe) the business environment?

Asking the First Questions

- final set of questions focuses on the effectiveness of the communication activity itself
  - Are you the right person to answer these questions?
  - Are your answers “official”?
  - Are my questions relevant to the problem that you have?
  - Am I asking too many questions?
  - Can anyone else provide additional information?
  - Should I be asking you anything else?
- A one or two page product request is prepared

- Sample product request written by a marketing person involved in the SafeHome project



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. It can be programmed by the homeowner, and will automatically telephone a monitoring agency when a situation is detected.

Eliciting Requirements

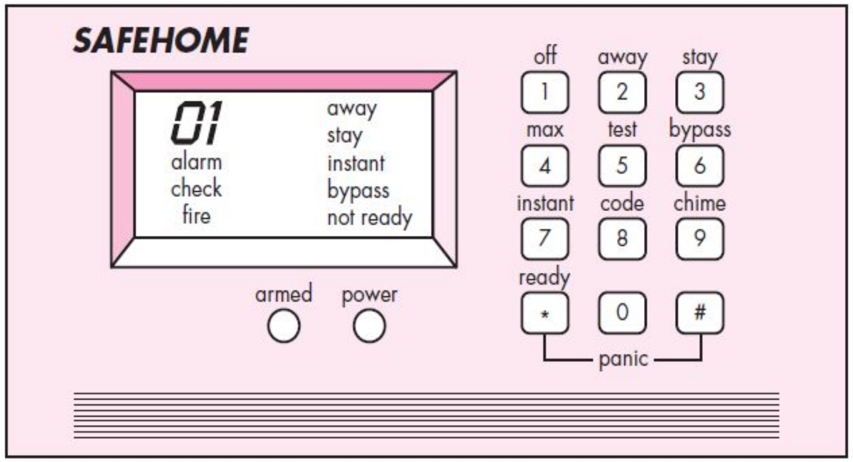
Collaborative Requirements Gathering

- Meetings are conducted and attended by
  - both software engineers and other stakeholders
- Rules for preparation and participation are established
- An agenda is suggested
- A “facilitator” controls the meeting
  - Customer
  - a developer,
  - or an outsider

- “definition mechanism” is used
    - work sheets
    - flip charts
    - wall stickers
    - an electronic bulletin board
    - chat room
    - virtual forum

- Meeting place, time and date are selected;
- A facilitator is chosen
- Product report is distributed to all attendees before the meeting date
- each attendee is asked to make a list of objects that
  - Surrounds the system
  - are produced by the system
  - are used by the system
- each attendee is asked to make another list of
  - Services (processes or functions)
  - Constraints(cost, size, business rules)
  - Performance criteria(speed, accuracy)

- Objects described for SafeHome might include
  - the control panel,
  - smoke detectors,
  - window and door sensors,
  - motion detectors,
  - an alarm,
  - an event (a sensor has been activated),
  - a display,
  - a PC,
  - telephone numbers,
  - a telephone call, and so on

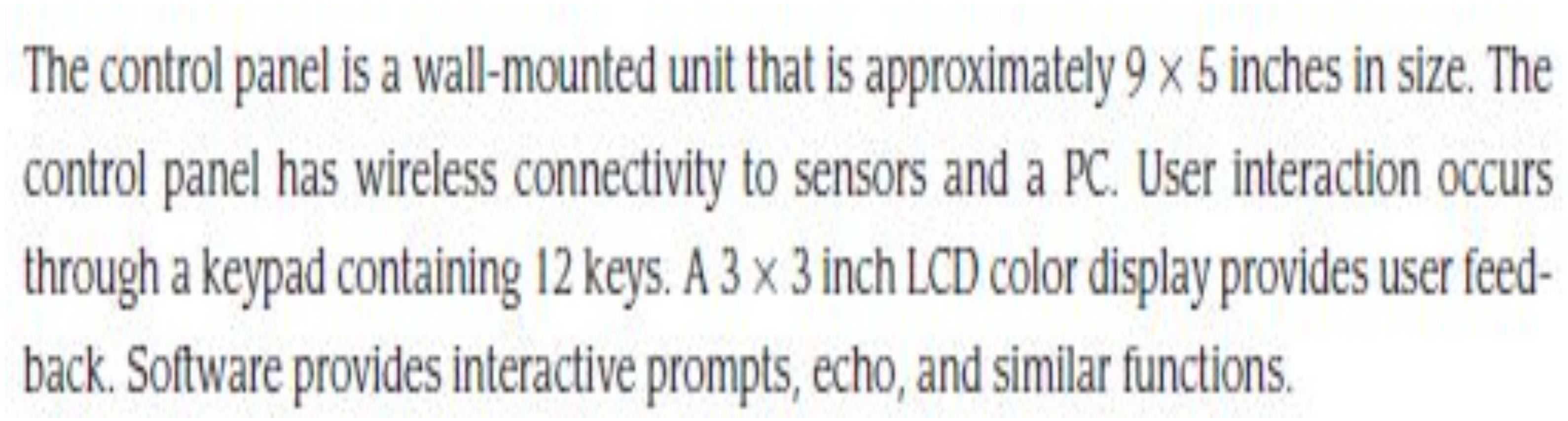


SafeHome control panel

- The list of services might include
    - configuring the system,
    - setting the alarm,
    - monitoring the sensors,
    - dialing the phone,
    - programming the control panel,
    - and reading the display

(note that services act on objects)

- lists of constraints
    - the system must recognize when sensors are not operating
    - must be user-friendly,
    - must interface directly to a standard phone line
- list of performance criteria
    - a sensor event should be recognized within one second
- stakeholders develop mini-specifications for entries on the lists


The control panel is a wall-mounted unit that is approximately 9 × 5 inches in size. The control panel has wireless connectivity to sensors and a PC. User interaction occurs through a keypad containing 12 keys. A 3 × 3 inch LCD color display provides user feed-back. Software provides interactive prompts, echo, and similar functions.

- The lists of objects can be
    - pinned to the walls(using large sheets of the paper)
    - Stuck to the walls(using adhesive back-sheets)
    - Written on a wall board
- Alternatively, the lists may have been posted
    - on an electronic bulletin board
    - at an internal website
    - or posed in a chat room
- Each list entry is capable of being manipulated separately
- List can be combined
- Entries can be modified
- Additions can be made
- At this stage, critique and debate are strictly prohibited

- After individual lists are presented
  - the group creates a combined list by
    - eliminating redundant entries,
    - adding any new ideas that come up during the discussion
  - but not deleting anything
- The combined list is
  - shortened,
  - lengthened,
  - reworded to properly reflect the product/system to be developed

- During all discussions
    - the team may raise an issue that cannot be resolved during the meeting
    - An issues list is maintained so that these ideas will be acted on later

Quality Function Deployment

- QFD is a quality management technique
- Translates the needs of the customer into technical requirements
- Concentrates on maximizing customer satisfaction

Normal requirements

- If these requirements are present, the customer is satisfied

Expected requirements

- Implicit requirements
- so fundamental
- Customer may not state them
- Their absence will be a cause for significant dissatisfaction

Exciting requirements

- features go beyond the customer’s expectations
- prove to be very satisfying when present

- QFD uses
  - customer interviews and observation
  - Surveys
  - examination of historical data

- These data are then translated into a table of requirements : customer voice table

- reviewed with the customer and other stakeholders

- variety of
  - diagrams,
  - matrices
  - evaluation methods are used
    - to extract expected requirements
    - to attempt to derive exciting requirements

Elicitation work products

- Work products depend on the size of the system or product to be built
    - Statement of need and feasibility
    - Bounded statement of scope for the system or product
    - List of customers, users and other stakeholders participated in the meeting
    - Description of system’s technical environment
    - List of requirements, services, constraints and performance criterions
    - Set of usage scenarios that provide insight into use of system
    - Prototypes developed to better define requirements

Agile Requirements Elicitation

- Requirements are elicited by asking all stakeholders to create ‘user stories’
- User story describes simple system requirement written from user’s perspective
- User stories written on small note cards
    - Easy for developers to select and manage subset of requirements
    - Allows user to communicate with stakeholders on selected requirements
- Problem
    - Consideration of business goals and non-functional requirements is lacking

- Usage Scenarios
  - it is difficult to move into more technical
  - developers and users can create set of scenarios to identify a thread of usage for the system
  - The scenarios, often called use cases
    - description of how the system will be used

Developing use cases

- Use case describes the behavior of the system under various condition
- tells a stylized story about how an end user interacts with the system
- The story may be
  - narrative text
  - Outline of tasks and interactions
  - Template based description
  - Diagrammatic representation
- Use case define set of “actors” involved in the story

- an actor is anything that communicates with the system
- Every actor has one or more goals when using the system
- Actor and end user are not necessarily same
- For instance,
  - Machine operator controlling robotics can act as
    - Programmer
    - Tester
    - Trouble shooter
- not all actors are identified during the first iteration
- It is possible to identify primary actors during the first iteration
- secondary actors as more is learned about the system

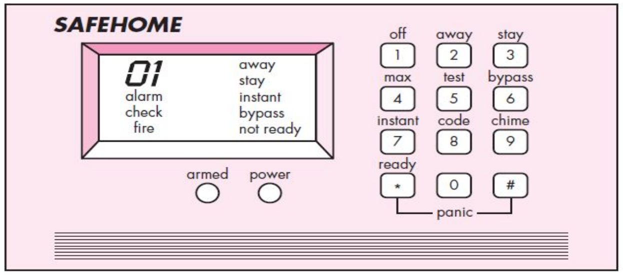
- Primary actors interact derive the intended benefit from the system
- They work directly and frequently with the software
- Secondary actors support the system so that primary actors can do their work
- Once actors have been identified, use cases can be developed

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?
- 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?

- Four actors of safe home project
  - Homeowner (a user)
  - Setup manager (likely the same person as home owner, but playing a different role)
  - Sensors (devices attached to the system)
  - Monitoring and response subsystem (central station)

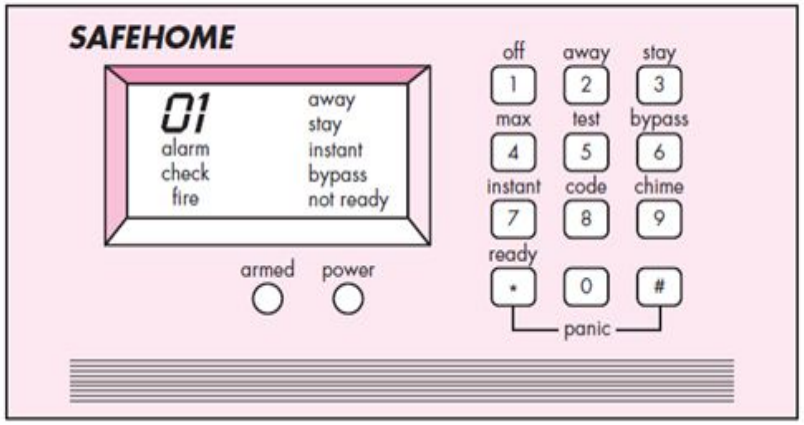
- homeowner actor interacts with the home security function in a number of different ways using
  - alarm control panel
  - a PC



- Enters a password to allow all other interactions
- Inquires about the status of a security zone
- Inquires about the status of a sensor
- Presses the panic button in an emergency
- Activates/deactivates the security system

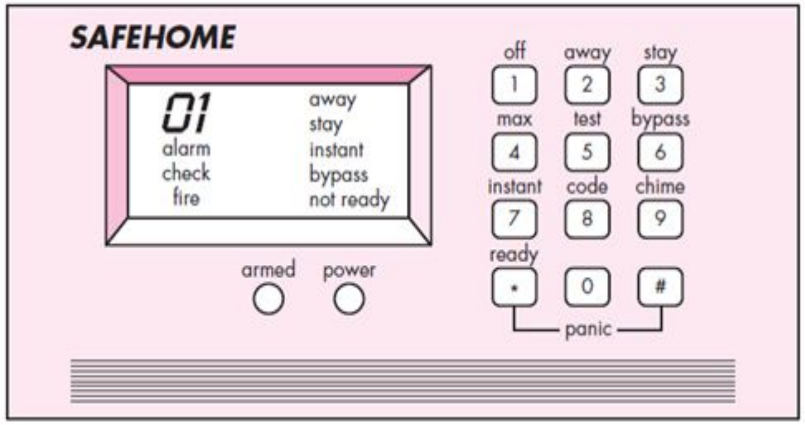
1.

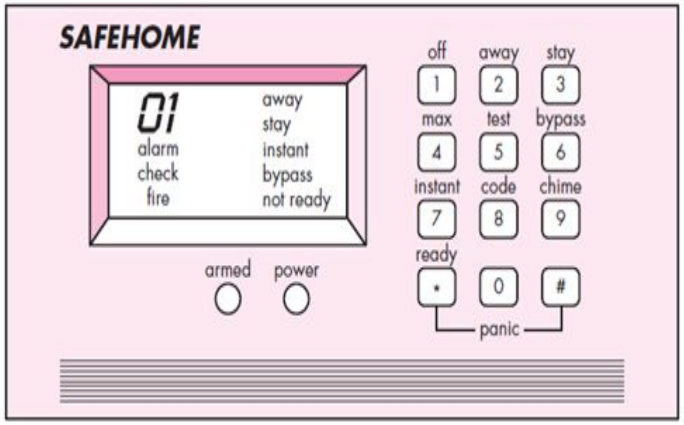
- The homeowner observes the SafeHome control panel to determine if the system is ready for input
- If the system is not ready, a not ready message is displayed on the LCD display
- the homeowner must physically close windows or doors so that the not ready message disappears



2.

- The homeowner uses the keypad to key in a four-digit password
- The password is compared with the valid password stored in the system
- If the password is incorrect, the control panel will beep once and reset itself for additional input
- If the password is correct, the control panel awaits further action





3.

- The homeowner selects and keys in stay or away to activate the system
- Stay activates only perimeter sensors (inside motion detection sensors are deactivated)
- Away activates all sensors

4.

- When activation occurs, a red alarm light can be observed by the homeowner

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:

1. Control panel is *not ready:* homeowner checks all sensors to determine which are open; closes them.

2. Password is incorrect (control panel beeps once): homeowner reenters correct password.

3. Password not recognized: monitoring and response subsystem must be contacted to reprogram password.

4. *Stay* is selected: control panel beeps twice and a *stay* light is lit; perimeter sensors are activated.

5. *Away* is selected: control panel beeps three times and an *away* light is lit; all sensors are activated.

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, sensors

Channels to secondary actors:

Support technician: phone line

Sensors: hardwired and radio frequency interfaces

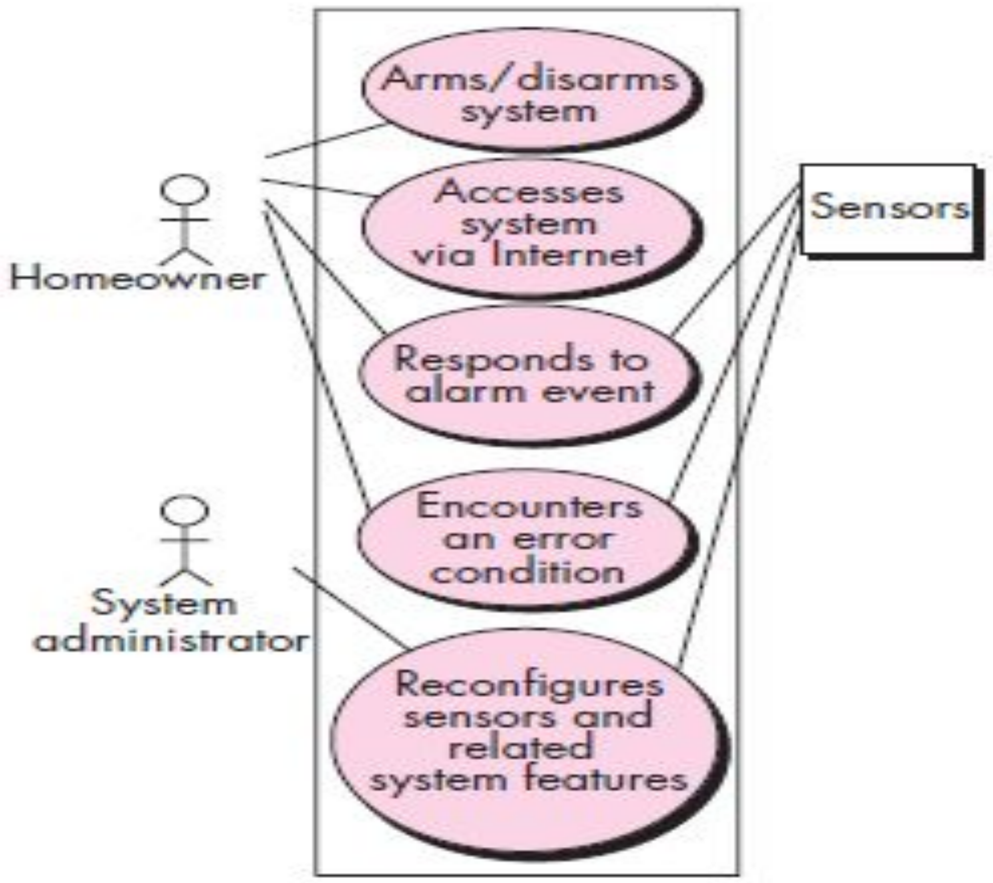
Open issues:

1. Should there be a way to activate the system without the use of a password or with an abbreviated password?

2. Should the control panel display additional text messages?

3. How much time does the homeowner have to enter the password from the time the first key is pressed?

4. Is there a way to deactivate the system before it actually activates?

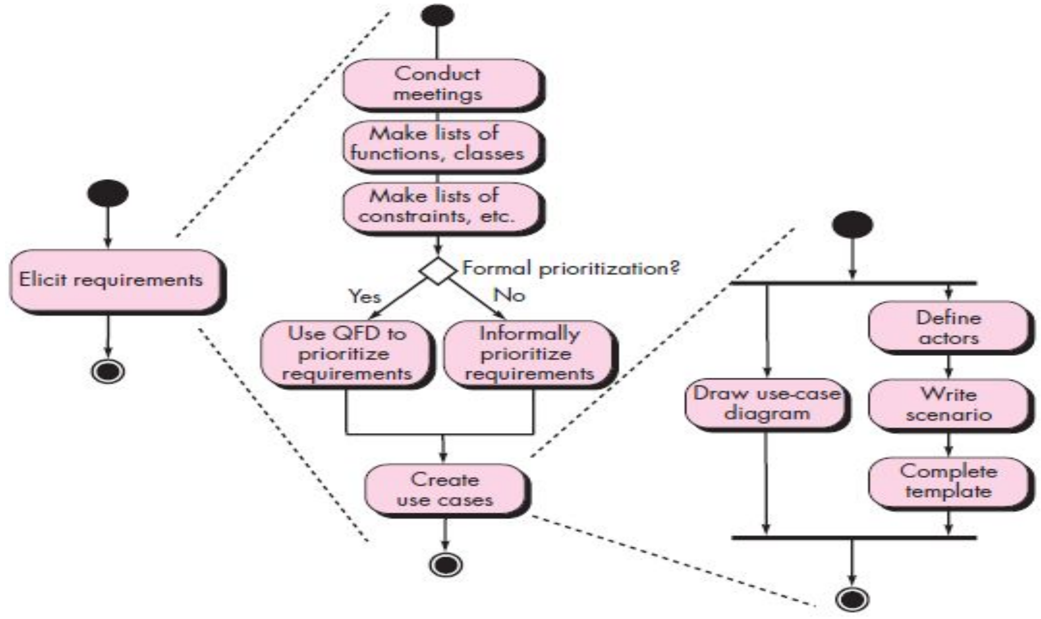

Arms/disarms system
Accesses system via Internet
Responds to alarm event
Encounters an error condition
Reconfigures sensors and related system features
Homeowner
System administrator
Sensors

Building Requirements Model

Scenario-based elements

use case

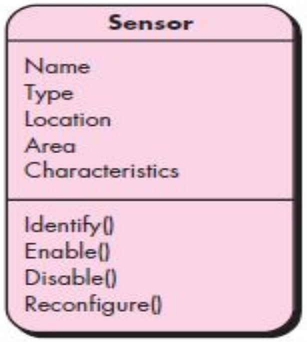
activity diagram



Class-based elements

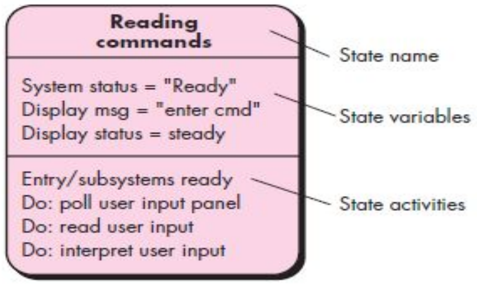
Class diagram

Class responsibility collaborator



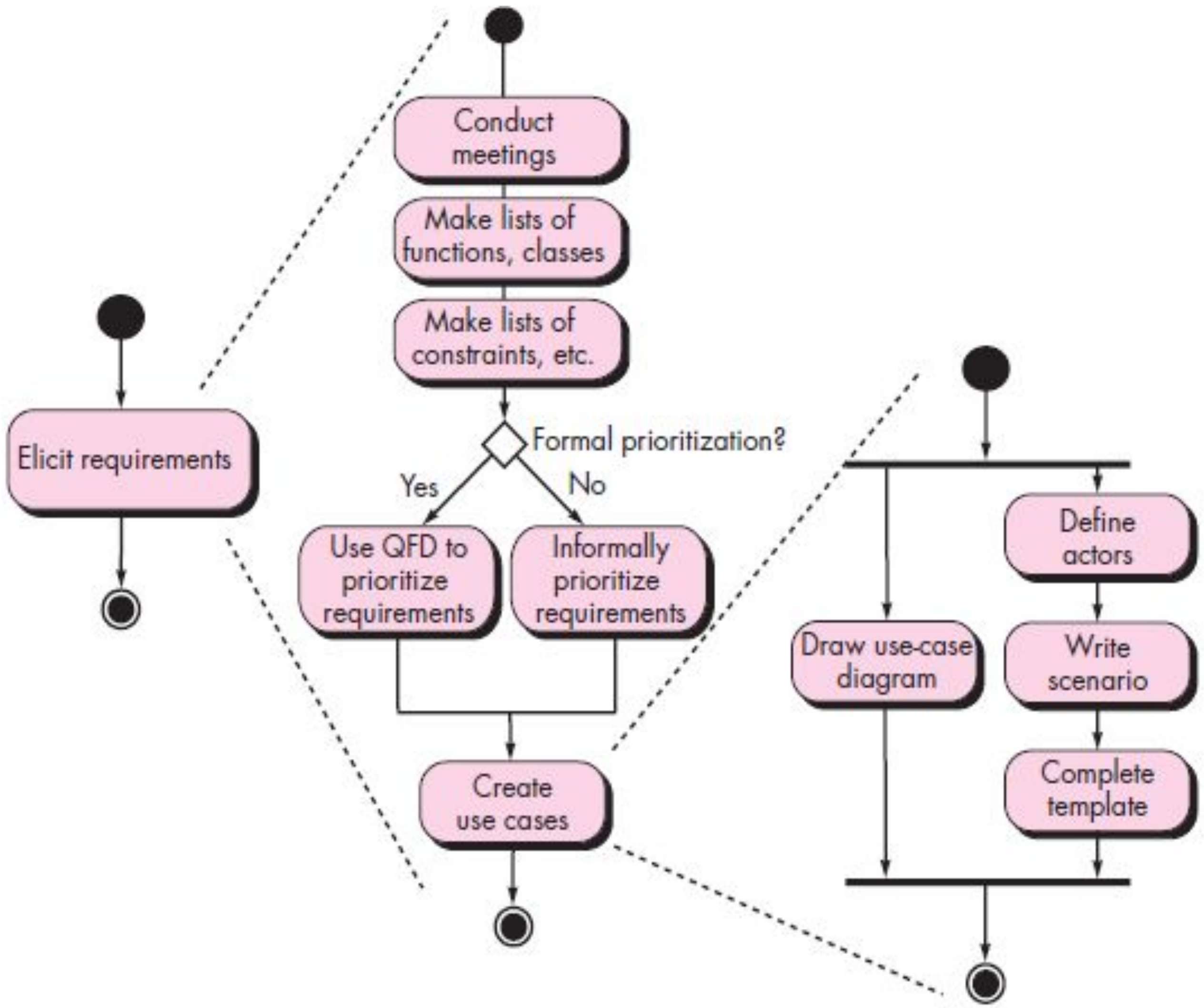
Behavioral elements

state diagram



Flow-oriented elements

data flow


Elicit requirements
Conduct meetings
Make lists of functions, classes
Make lists of constraints, etc.
Formal prioritization?
Yes
No
Use QFD to prioritize requirements
Informally prioritize requirements
Create use cases
Define actors
Draw use-case diagram
Write scenario
Complete template

Negotiating requirements

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

Validating requirements

1. Do any requirements conflict with other requirements?

2. Is each requirement bounded and unambiguous?

3. Is each requirement testable, once implemented?

4. Have all requirements been specified at the proper level of abstraction?

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

Analysis Patterns

- These analysis patterns suggest solutions within the application domain that can be reused when modeling many applications
- analysis patterns speed up the development of abstract analysis models
- Analysis patterns facilitate the transformation of the analysis model into a design model