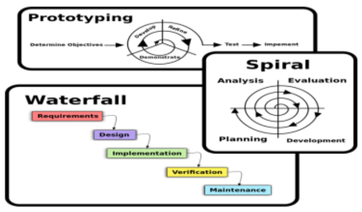
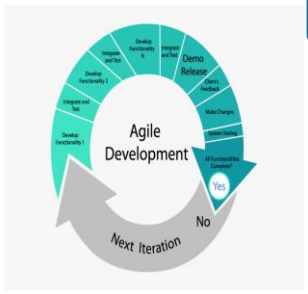
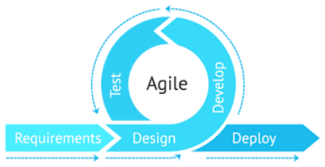


Software Engineering Understanding Requirements





Outline

- Requirements Engineering
- Establishing the Ground Work
- Eliciting Requirements
- Developing Use Cases
- Building the Requirement Model
- Negotiating Requirements
- Validating Requirements.

Introduction

- Understanding the requirements of a problem is among the most difficult tasks that face a software engineer.
- When you first think about it, developing a clear understanding of requirements doesn't seem that hard. as we think - the customer knows what is required, he has a good understanding of the features and functions that will provide benefit
- Surprisingly, in many instances the answer to these is not the case.
- And even if customers and end-users are explicit in their needs, those needs will change throughout the project.
- A customer walks into your office, sits down, looks you straight in the eye, and says,
- "I know you think you understand what I said, but what you don't understand is what I said is not what I meant."
- Invariably, this happens late.

Introduction

- **What is it?**
  - set of requirements engineering tasks - to understand the business impact of software, customer needs, and users interaction with the software.
- **Who does it?**
  - Software engineers (system engineers /analysts) and other project stakeholders (managers, customers, end users)
- **Why is it important?**
  - understand what the customer wants before you begin to design and build a computer-based system.

Introduction

- **What are the steps?**
  - inception:(scope and nature) -negotiation:
  - elicitation: (define what is required,) - specification and validation
  - elaboration: (refined and modified.)
- **What is the work product?**
  - written understanding of the problem. usage scenarios, functions and features lists, requirements models, or a specification.
- **How do I ensure that I've done it right?**
  - reviewed with stakeholders

5

Requirement Engineering

- Building software is so compelling that many software developers want to jump right in before they have a clear understanding of what is needed.
- Things change so rapidly that any attempt to understand requirements in detail is a waste of time, that the bottom line is producing a working program and all else is secondary. (can be true for small projects)
- 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.

6

Requirement Analysis is Hard

“The **hardest** single **part** of building a software system is **deciding what to build**. No part of the work so cripples the resulting system if done wrong.”

"Fred" Brooks Jr. is an American computer architect, software engineer, and computer scientist, best known for managing the development of IBM's System/360 family of computers

“The **seeds** of major software **disasters** are usually **sown** in the **first three months** of commencing the software project.”

Capers Jones is an American specialist in software engineering methodologies

7

Requirement Analysis is Hard

What is Requirement Engineering?

Tasks and **techniques** that lead to an **understanding** of **requirements** is called **requirement engineering**.

8

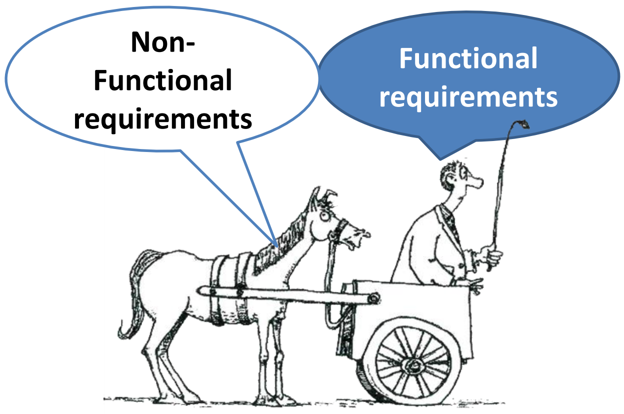
Requirement Engineering

- It must be adapted to the needs of the process, the project, the product, and the people doing the work.
- Requirements engineering builds a bridge to design and construction.
- Requirements engineering provides the appropriate mechanism for understanding
  - what the customer wants,
  - analyzing need,
  - assessing feasibility,
  - negotiating a reasonable solution,
  - specifying the solution unambiguously,
  - validating the specification, and
  - managing the requirements as they are transformed into an operational system

9

Requirement Falls in two types

Functional requirements	Non-Functional requirements



Don't put what you want to do - before how you need to do it

10

Functional Requirement

- Any requirement which specifies **what the system Should do**.
- A functional requirement will describe a particular behaviour of function of the system when certain conditions are met, for example: "Send email when a new customer signs up" or "Open a new account".

Typical functional requirements

- Business Rules
- Administrative functions
- Historical Data
- Transaction corrections, adjustments and cancellations
- Legal or Regulatory Requirements
- External Interfaces
- Reporting Requirements
- Authentication
- Authorization levels
- Audit Tracking

Non-Functional Requirement

- Any requirement which specifies **how the system performs a certain function**.
- A non-functional requirement will describe how a system should behave and what limits there are on its functionality.

Typical non-functional requirements

Response time	Availability	Regulatory
Throughput	Reliability	Manageability
Utilization	Recoverability	Environmental
Static volumetric	Maintainability	Data Integrity
Scalability	Serviceability	Usability
Capacity	Security	Interoperability

Library Management System

- **Functional Requirement:**
  - **Add Article:** New entries must be entered in database
  - **Update Article:** Any changes in articles should be updated in case of update
  - **Delete Article:** Wrong entry must be removed from system
  - **Inquiry Members:** Inquiry all current enrolled members to view their details
  - **Inquiry Issuance:** Inquiry all database articles
  - **Check out Article:** To issue any article must be checked out
  - **Check In article:** After receiving any article system will reenter article by Checking
  - **Inquiry waiting for approvals:** Librarian will generates all newly application which is in waiting list
  - **Reserve Article:** This use case is used to reserve any book with the name of librarian, it can be pledged

Library Management System

- **Non-Functional Requirement:**
  - **Safety Requirements:** The database may get crashed at any certain time due to virus or operating system failure. So, it is required to take the database backup.
  - **Security Requirements:** We are going to develop a secured database for the university. There are different categories of users namely teaching staff, administrator, library staff, students etc., Depending upon the category of user the access rights are decided.
  - **Software Constraints:** The development of the system will be constrained by the availability of required software such as database and development tools. The availability of these tools will be governed by.

Requirement Engineering

- It 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.

15

Requirement Engineering Tasks

1 Inception

- Roughly define scope
- A basic understanding of a problem, people who want a solution, the nature of solution desired

2 Elicitation **(Requirement Gathering)**

- Define requirements
- The practice of collecting the requirements of a system from users, customers and other stakeholders

16

Requirement Engineering Tasks

3 Elaboration

- Further define requirements
- Expand and refine requirements obtained from inception & elicitation
- Creation of User scenarios, extract analysis class and business domain entities

4 Negotiation

- Reconcile conflicts
- Agree on a deliverable system that is realistic for developers and customers

17

Requirement Engineering Tasks

5 Specification

- Create analysis model
- It may be written document, set of graphical models, formal mathematical model, collection of user scenarios, prototype or collection of these
- **SRS (Software Requirement Specification)** is a document that is created when a detailed description of all aspects of software to build must be specified before starting of project

6 Validation

- Ensure quality of requirements
- Review the requirements specification for errors, ambiguities, omissions (absence) and conflicts

18

Requirement Engineering Tasks

7 Requirements Management

- It is a set of activities to identify, control & trace requirements & changes to requirements (Umbrella Activities) at any time as the project proceeds.

Inception

- How does a software project get started? Is there a single event that becomes the catalyst for a new computer-based system or product, or does the need evolve over time?
- There are no definitive answers to these questions. In some cases, a casual conversation is all that is needed to precipitate a major software engineering effort.
- But in general, most projects begin when a business need is identified or a potential new market or service is discovered.
- Stakeholders from the business community (e.g., business managers, marketing people, product managers) define a business case for the idea, try to identify the breadth and depth of the market, do a rough feasibility analysis, and identify a working description of the project's scope.

Inception

- All of this information is subject to change, but it is sufficient to precipitate discussions with the software engineering organization.
- 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.

21

Eliciting is the hardest part!

- Elicitation is the Hardest Part!
- **Problems of scope**
  - System boundaries are ill-defined
  - Customers will provide irrelevant information
- **Problems of understanding**
  - Customers never know exactly what they want
  - Customers don't understand capabilities and limitations
  - Customers have trouble fully communicating needs
- **Problems of volatility**
  - Requirements always change

22

Eliciting

- It certainly seems simple enough
  - Ask the customer, the users, and others what the objectives for the system or product are,
  - what is to be accomplished,
  - how the system or product fits into the needs of the business, and
  - finally, how the system or product is to be used on a day-to-day basis.
- But it isn’t simple—it’s very hard.
- Number of problems that are encountered as elicitation occurs.
  - **Problems of scope**: The boundary of the system is ill-defined or the customers/users specify unnecessary technical detail that may confuse, rather than clarify, overall system objectives.

23

Eliciting

  - **Problems of understanding**: The customers/users are not completely sure of what is needed, have a poor understanding of the capabilities and limitations of their computing environment, don’t have a full understanding of the problem domain, have trouble communicating needs to the system engineer, omit information that is believed to be “obvious,” specify requirements that conflict with the needs of other customers/users, or specify requirements that are ambiguous or untestable.
  - **Problems of volatility:** The requirements change over time.
- To help overcome these problems, you must approach requirements gathering in an organized manner.

24

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 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.

Negotiation

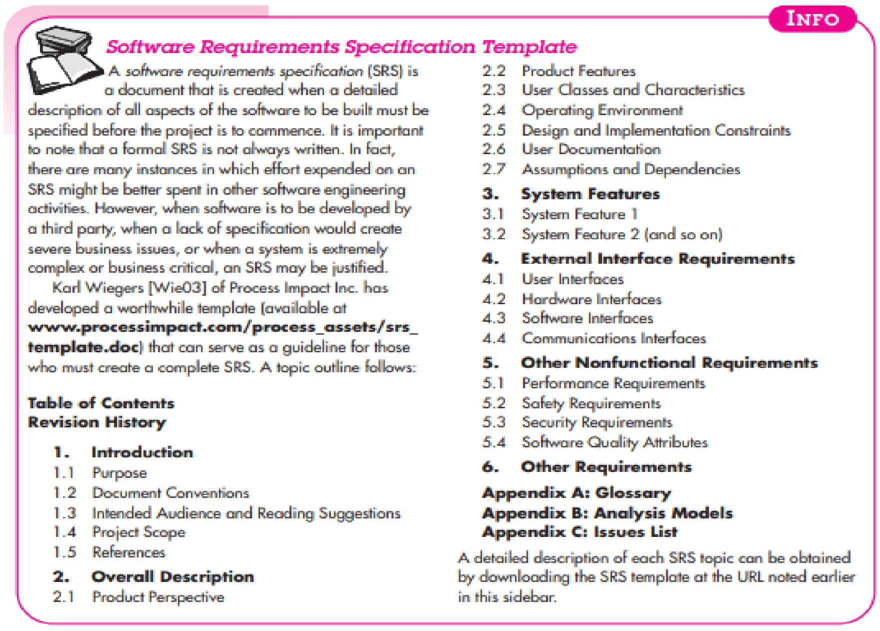
- It isn't unusual for customers and users to ask for more than can be achieved, given limited business resources.
- It's also relatively common for different customers or users to propose conflicting requirements, arguing that their version is "essential for our special needs."
- You have to resolve 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.

Specification

- In the context of computer-based systems (and software), the term specification means different things to different people.
- A specification 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.
- Some suggest that a "standard template" should be developed and used for a specification, arguing that this leads to requirements that are presented in a consistent and therefore more understandable manner.
- It is sometimes necessary to remain flexible when a specification is to be developed. large systems requires a written document, combining natural language descriptions and graphical models. However, usage scenarios may be all that are required for smaller products or systems that reside within well-understood technical environments.

27

Software Requirement Specification



INFO

Software Requirements Specification Template

A software requirements specification (SRS) is a document 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 earlier in this sidebar.

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.
- The primary requirements validation mechanism is the technical review.

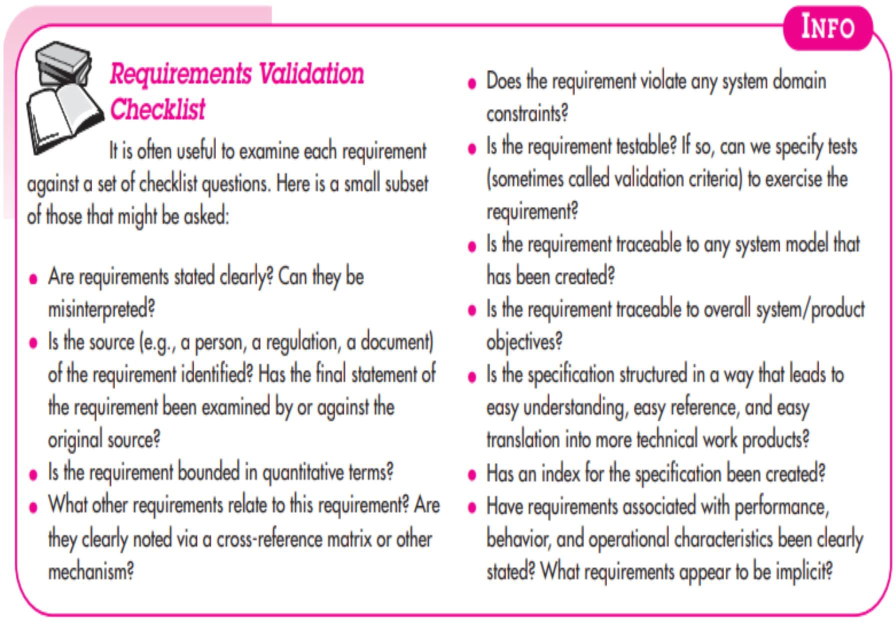
29

Validation

- 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, conflicting requirements, or unrealistic (unachievable) requirements.

30

Validation


INFO

Requirements Validation Checklist

It is often useful to examine each requirement against a set of checklist questions. Here is a small subset of those that might be asked:

- Are requirements stated clearly? Can they be misinterpreted?
- Is the source (e.g., a person, a regulation, a document) of the requirement identified? Has the final statement of the requirement been examined by or against the original source?
- Is the requirement bounded in quantitative terms?
- What other requirements relate to this requirement? Are they clearly noted via a cross-reference matrix or other mechanism?
- Does the requirement violate any system domain constraints?
- Is the requirement testable? If so, can we specify tests (sometimes called validation criteria) to exercise the requirement?
- Is the requirement traceable to any system model that has been created?
- Is the requirement traceable to overall system/product objectives?
- Is the specification structured in a way that leads to easy understanding, easy reference, and easy translation into more technical work products?
- Has an index for the specification been created?
- Have requirements associated with performance, behavior, and operational characteristics been clearly stated? What requirements appear to be implicit?

Requirement 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 activities are identical to the software configuration management (SCM) techniques. [covered later stage in book]
- Formal requirements management is initiated only for large projects that have hundreds of identifiable requirements. For small projects, this requirements engineering action is considerably less formal.

32

Establishing the Ground Work

- Stakeholders and software engineers work together on the same team. In such cases, requirements engineering is simply a matter of conducting meaningful conversations with colleagues who are well-known members of the team.
- But reality is often quite different.
- Customer(s) or end users:
  - may be located in a different city or country,
  - may have only a vague idea of what is required,
  - may have conflicting opinions about the system to be built,
  - may have limited technical knowledge, and
  - may have limited time to interact with the requirements engineer.
- None of these things are desirable, but all are fairly common, and you are often forced to work within the constraints imposed by this situation

33

Project Inception

- During the **initial project meetings**, the following tasks should be accomplished
- **Identify the project stakeholders**
  - These are the folks we should be talking to
- **Recognize multiple viewpoints**
  - Stakeholders may have different (and conflicting) requirements
- **Work toward collaboration**
  - It's all about reconciling conflict
- **Ask the first questions**
  - Who? What are the benefits? Another source?
  - What is the problem? What defines success? Other constraints?
  - Am I doing my job right?

34

Establishing the Ground Work

- Following are the steps required to establish the groundwork for an understanding of software requirements—to get the project started in a way that will keep it moving forward toward a successful solution.
  - Identifying Stakeholders
  - Recognizing Multiple Viewpoints
  - Working toward Collaboration
  - Asking the First Questions

35

Identifying Stakeholders

- Stockholders may be business operations managers, product managers, marketing people, internal and external customers, end users, consultants, product engineers, software engineers, support and maintenance engineers, and others.
- Each stakeholder has a different view of the system, achieves different benefits when the system is successfully developed, and is open to different risks if the development effort should fail.
- At inception, a list is created of people who will contribute input as requirements are elicited.
- The initial list will grow as stakeholders are contacted because every stakeholder will be asked: “Whom else do you think I should talk to?”

36

Recognizing Multiple Viewpoints

- Because many different stakeholders exist, the requirements of the system will be explored from many different points of view.
- For example, the marketing group is interested in functions and features that will excite the potential market, making the new system easy to sell.
- Business managers are interested in a feature set that can be built within budget and that will be ready to meet defined market windows.

Working toward Collaboration

- The job of a requirements engineer is to identify areas of commonality (i.e., requirements on which all stakeholders agree) and areas of conflict or inconsistency (i.e., requirements that are desired by one stakeholder but conflict with the needs of another stakeholder).
- Collaboration does not necessarily mean that requirements are defined by committee.
- In many cases, stakeholders collaborate by providing their view of requirements, but a strong “project champion”(e.g., a business manager or a senior technologist) may make the final decision about which requirements make the cut.

Asking the First Questions

- Questions asked at the inception of the project should be "context free".
- The first set of context-free questions focuses on the customer and other stakeholders, the overall project goals and benefits.
- For example, you might ask:
  - 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?
- These questions help to identify all stakeholders who will have interest in the software to be built.
- In addition, the questions identify the measurable benefit of a successful implementation and possible alternatives to custom software development.

Asking Other Questions

- The next set of questions enables you to gain a better understanding of the problem and allows the customer to voice his or her perceptions about a solution:
  - How would you characterize "good" output that would be generated by a successful solution?
  - What problem(s) will this solution address?
  - Can you show me (or describe) the business environment in which the solution will be used?
  - Will special performance issues or constraints affect the way the solution is approached?

Asking Final Questions

- The 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?
- These questions (and others) will help to "break the ice" and initiate the communication that is essential to successful elicitation.

41

Eliciting Requirements

- Requirements elicitation (also called requirements gathering) combines elements of <u>problem solving</u>, <u>elaboration</u>, <u>negotiation</u>, and <u>specification</u>.
- In order to encourage a collaborative, team-oriented approach to requirements gathering, <u>stakeholders</u> work together to <u>identify the problem</u>, <u>propose elements</u> of the solution, negotiate <u>different approaches</u> and specify a <u>preliminary set of solution requirements</u>.
- This approach is sometimes called a facilitated application specification technique (FAST).

42

Collaborative Requirement Gathering

- Many different approaches to collaborative requirements gathering have been proposed.
- Each makes use of a slightly different scenario, but all apply some variation on the following basic guidelines:
  - Meetings are conducted and attended by both software engineers and other stakeholders.
  - Rules for preparation and participation are established.
  - An agenda is suggested that is formal enough to cover all important points but informal enough to encourage the free flow of ideas.
  - 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.

Collaborative Requirement Gathering

- The goal is to identify the problem, propose elements of the solution, negotiate different approaches, and specify a preliminary set of solution requirements in an atmosphere that is conducive/ possible to the accomplishment of the goal.
- For Eg. during inception basic questions and answers establish the scope of the problem and the overall perception of a solution.
- Out of these initial meetings, the developer and customers write a one- or two-page "product request."
- 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.

Collaborative Requirement Gathering

- As an example, let’s consider an excerpt from a product request written by a marketing person involved in the SafeHome project.
- This person writes the following narrative about the home security function that is to be part of SafeHome:
  - 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.

Collaborative Requirement Gathering

  - It can be programmed by the homeowner, and will automatically telephone a monitoring agency when a situation is detected.
- All would contribute to this narrative during the requirements gathering meeting and considerably more information would be available.
- But even with additional information,
  - ambiguity would be present,
  - omissions would likely exist, and
  - errors might occur.
- For now, the defined “functional description” will suffice.

Collaborative Requirement Gathering

- each attendee is asked to make a list of (before meeting)
  - objects that are part of the environment that surrounds the system,
  - other objects that are to be produced by the system, and
  - objects that are used by the system to perform its functions.
- In addition, each attendee is asked to make another list of services (processes or functions) that manipulate or interact with the objects.
- Finally, lists of constraints (e.g., cost, size, business rules) and performance criteria (e.g., speed, accuracy) are also developed.
- lists are not expected to be exhaustive but are expected to reflect each person's perception of the system.

47

Collaborative Elicitation

One-on-one Q&A sessions rarely succeed in practice; collaborative strategies are more practical

48

Elicitation work products

- Collaborative elicitation should result in several **work products**
  - A bounded statement of scope
  - A list of stakeholders
  - A description of the technical environment
  - A list of requirements and constraints
  - Any prototypes developed
  - A set of use cases
- Characterize how users will interact with the system
- Use cases tie functional requirements together

49

Collaborative Requirement Gathering

- 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.

50

Collaborative Requirement Gathering

- 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).
- In a similar fashion, each attendee will develop lists of constraints e.g.,
  - the system must recognize when sensors are not operating,
  - must be user-friendly,
  - must interface directly to a standard phone line) and
  - performance criteria (e.g., a sensor event should be recognized within one second, and an event priority scheme should be implemented).
- The objective is to develop a consensus list of objects, services, constraints, and performance for the system to be built

Quality Function Deployment (QFD)

- Quality function deployment (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.
- QFD identifies three types of requirements :
  - **Normal requirements**: 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.
  - Examples of normal requirements might be requested types of graphical displays, specific system functions, and defined levels of performance.

Quality Function Deployment (QFD)

- **Expected requirements**: These 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.
- Examples of expected requirements are: ease of human/machine interaction, overall operational correctness and reliability, and ease of software installation.
- **Exciting requirements**: These features go beyond the customer's expectations and prove to be very satisfying when present.
- For example, software for a new mobile phone comes with standard features, but is coupled with a set of unexpected capabilities (e.g., multitouch screen, visual voicemail) that delight every user of the product.

Quality Function Deployment (QFD)

- 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 requirement

Quality Function Deployment (QFD)

- This is a technique that **translates** the **needs** of the **customer** into **technical requirements** for software
- It **emphasizes** an understanding of **what is valuable to the customer** and then deploys these values throughout the engineering process through functions, information, and tasks
- It **identifies** three types of **requirements**
  - **Normal requirements:** These requirements are the objectives and goals stated for a product or system during meetings with the customer
  - **Expected requirements:** These requirements are implicit to the product or system and may be so fundamental that the customer does not explicitly state them
  - **Exciting requirements:** These requirements are for features that go beyond the customer's expectations and prove to be very satisfying when present