REQUIREMENT ENGINEERING INTRODUCTION

(A business analyst approach perspective)

REQUIREMENT ENGINEERING | Engenharia de Requisitos

2023/24

Agenda

- Requirement Sources
- Inception
- Elicitation
- Elaboration
- Negotiation
- Specification
- Validation
- Management

Requirement sources

Requirement Sources / Find Problems

- **Análise de concorrência**
- **Análise de diferenciação**
- **Dinâmica de Mercado *(Market Dynamics)* e investigação**
- **Investigação de utilizador**
- **Resposta ou Reação *(Feedback)***
- **Criatividade**

Requirement Sources / Find Problems

- Introspeção
- Mineração de dados *(Data mining)*
- Análise do sistema atual
- Monitorização do sistema
- Análise técnica
- Análise de documentação

Inception

The problem
Definition
Inception
Elicitation
Elaboration
Negotiation
Specification
Validation
Management

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.

Inception

- Identifying Stakeholders
- Recognizing Multiple Viewpoints
  - Ex. Marketing group, Business managers, etc.
- Working toward Collaboration
  - Identify areas of commonality and areas of conflict or inconsistency (i.e., requirements that are desired by one stakeholder but conflict with the needs of another stakeholder)

Inception - Asking the First Questions

The first set of context-free questions focuses on the customer and other stakeholders, the overall project goals and benefits. Example:

- 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?

Inception – Next set of questions

The next set of questions enables you to gain a better understanding of the problem and allows the customer to voice his perceptions about a solution. Example:

- 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?

Inception – “Final” set of questions

The “final” set of questions focuses on the effectiveness of the communication activity itself. Example:

- 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?

The Business Analyst

- Is the individual who has the primary responsibility to elicit, analyze, document and validate the needs of the stakeholders.
- The BA plays a central role in collecting and disseminating product information.
- Business analyst is a project role, not necessarily a job title.

Elicitation

The problem
Definition
Inception
Elicitation
Elaboration
Negotiation
Specification
Validation
Management

Elicitation

Requirements elicitation combines elements of **problem solving, elaboration, negotiation and specification**.

- encourage a collaborative, team-oriented approach to requirements gathering;

- stakeholders work together to identify the problem;

- propose elements of the solution;

- negotiate different approaches;

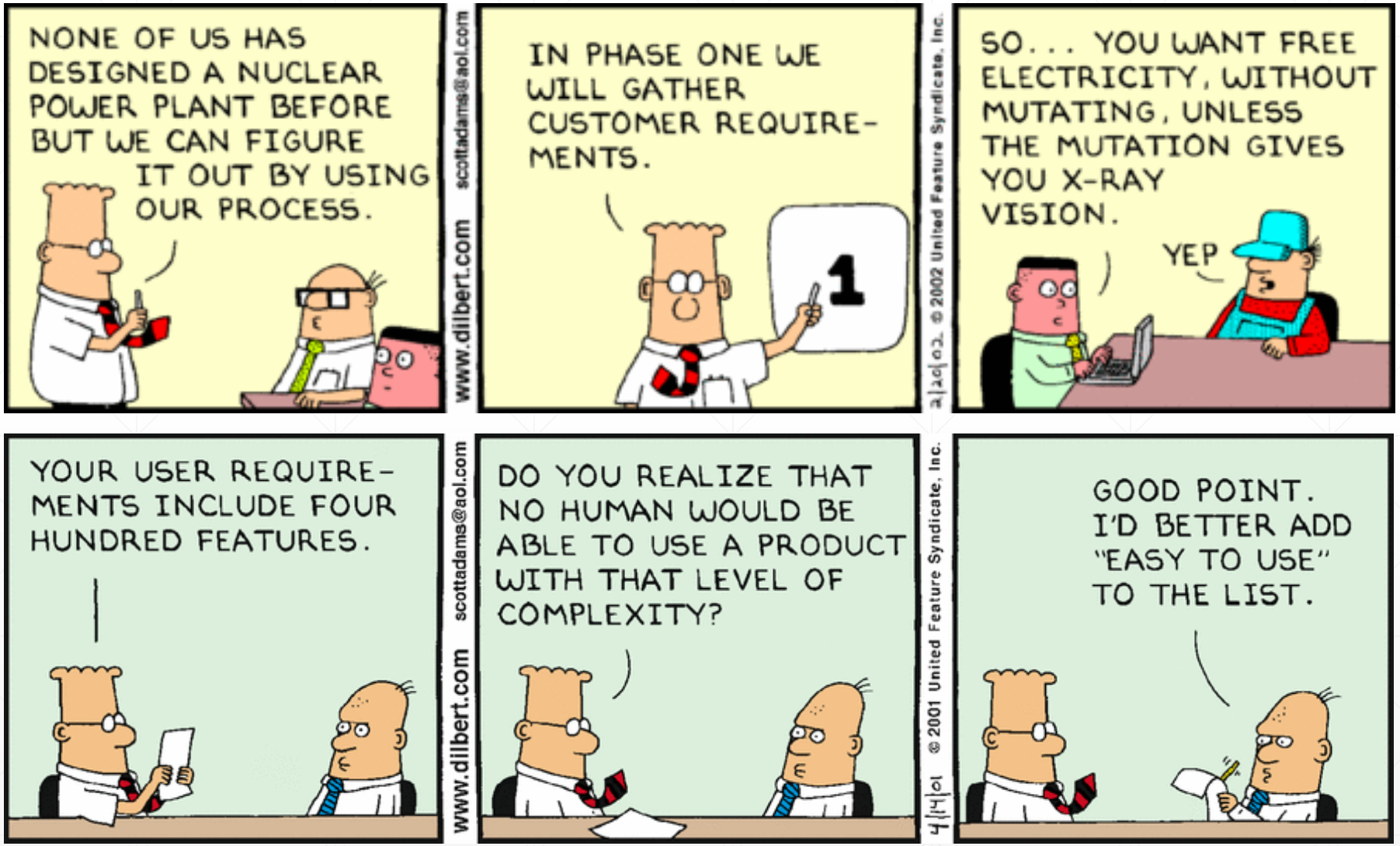
- specify a **preliminary set of solution requirements**.

Elicitation – Collaborative requirements

A basic guideline:

- 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.

Elicitation – Collaborative requirements



Elicitation – Collaborative requirements

- As an example, consider an excert 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:*

Elicitation – Collaborative requirements

"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."

Elicitation – Collaborative requirements

SafeHome

Conducting a Requirements Gathering Meeting

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; and a facilitator.

The conversation:

Facilitator (pointing at whiteboard): 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 that . . . 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 20 minutes refining and expanding the details of the home security function.)

Source: Software Engineering Practitioner's Approach

Elicitation – QFD

Quality function deployment (QFD) is a quality management technique that translates the needs of the customer into technical requirements for software:

- **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.
- **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.
- **Exciting requirements.** These features go beyond the customer's expectations

Elicitation – Usage Scenarios

- As requirements are gathered, an **overall vision** of system functions and features begins to materialize.
- 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.
- Developers and users can create a set of **scenarios** that identify a thread of usage for the system to be constructed. The scenarios, often called *use cases*, provide a description of how the **system will be used.**

Elicitation – Usage Scenarios

SafeHome

Developing a Preliminary User Scenario

The scene: A meeting room, continuing the first requirements gathering meeting.

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; and a facilitator.

The conversation:

Facilitator: We've been talking about security for access to *SafeHome* functionality that will be accessible via the Internet. I'd like to try something. Let's develop a usage scenario for access to the home security function.

Jamie: How?

Facilitator: We can do it a couple of different ways, but for now, I'd like to keep things really informal. Tell us (he points at a marketing person) how you envision accessing the system.

Marketing person: Um . . . well, this is the kind of thing I'd do if I was away from home and I had to let someone into the house, say a housekeeper or repair guy, who didn't have the security code.

Facilitator (smiling): That's the reason you'd do it . . . tell me how you'd actually do this.

Marketing person: Um . . . the first thing I'd need is a PC. I'd log on to a website we'd maintain for all users of *SafeHome*. I'd provide my user id and . . .

Vinod (interrupting): The Web page would have to be secure, encrypted, to guarantee that we're safe and . . .

Facilitator (interrupting): That's good information, Vinod, but it's technical. Let's just focus on how the end user will use this capability. OK?

Vinod: No problem.

Marketing person: So as I was saying, I'd log on to a website and provide my user ID and two levels of passwords.

Source: Software Engineering Practitioner's Approach

Elicitation – Usage Scenarios

Jamie: What if I forget my password?

Facilitator (interrupting): Good point, Jamie, but let's not address that now. We'll make a note of that and call it an *exception*. I'm sure there'll be others.

Marketing person: After I enter the passwords, a screen representing all *SafeHome* functions will appear. I'd select the home security function. The system might request that I verify who I am, say, by asking for my address or phone number or something. It would then display a picture of the security system control panel along with a list of functions that I can perform—arm the system, disarm the system, disarm one or more sensors. I suppose it might also allow me to reconfigure security zones and other things like that, but I'm not sure.

(As the marketing person continues talking, Doug takes copious notes; these form the basis for the first informal usage scenario. Alternatively, the marketing person could have been asked to write the scenario, but this would be done outside the meeting.)

Source: Software Engineering Practitioner's Approach

Elicitation techniques

Elicitation techniques
Interviews

- Asking users: most obvious way to find out what they need
- Mechanism to get direct user involvement
- Appropriate for “busy” executives
- Questions should be carefully prepared in advance

Note: include all the different users of the system

Elicitation techniques
Questionnaires

- Way to survey large groups of users to understand their needs
- Inexpensive, geographically independent
- Also used for feedback about products
- Biggest challenge: preparing well-written questions

Elicitation techniques Workshops

- Meeting with multiple stakeholders
- Encourage collaboration in defining requirements concurrently
- Needs a facilitator (critical)
- Can be resource intensive

"HOW TO TORTURE A BUSINESS ANALYST # 3"

Elicitation techniques
Focus groups

- Representative group of users
- Must be interactive: chance for all users to voice their thoughts
- Useful for exploring users' attitudes, impressions, preference and needs
- Must be facilitated
- Subjective feedback – need further evaluation

Elicitation techniques
Observations

- Observe how users perform their tasks
- Time consuming: time should be limited
- Important or high-risk tasks should be selected
- Can be silent (users cannot be interrupted) or interactive (asking questions allowed)
- Observed information should be documented for further analysis

Elicitation techniques
Document analysis

- Examining existing documentation
- Reduces the elicitation meeting time needed
- Can reveal information people don´t tell
- Risk: documents outdated

Elicitation techniques
User interface analysis

- Studying existing systems to discover user and functional requirements
- Uses screen shots if no direct interaction is possible
- Helps understanding how an existing system works

Elicitation techniques
System interface analysis

- Examining the systems to which your system connects
- It reveals functional requirements regarding data and services exchange between systems
- Identifying functionalities that may lead to requirements

Elicitation techniques Summary

- Interviews
- Workshops
- Focus groups
- Observation
- Document analysis
- User interface analysis
- System interface analysis

Elicitation – Work Products

The work products depends on the size of the system or product. It may include:

- 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.

Elicitation – Work Products (cont.)

- A list of requirements (preferably organized by function) and the domain constraints that apply to each.
- A set of preliminary usage scenarios that provide insight into the use of the system or product under different operating conditions.
- Any prototypes developed to better define requirements.

Elicitation Plan

- Elicitation objectives
- Elicitation strategy and planned techniques
- Schedule and resource estimates
- Documents and system needed for each elicitation
- Expected products
- Elicitation risks

Planning for elicitation

- Plan session scope and agenda
- Prepare resources
- Learn about the stakeholders
- Prepare questions
- Prepare analysis models

Elaboration

The problem
Definition
Inception
Elicitation
Elaboration
Negotiation
Specification
Validation
Management

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 actors will interact with the system.

Elaboration – used elements

- **Scenario-based elements.** The system is described from the user's point of view using a scenario-based approach. For example, basic use cases.
- **Class-based elements.** Each usage scenario implies a set of objects that are manipulated as an actor interacts with the system. These objects are categorized into classes.

Elaboration – used elements

- **Behavioral elements.** 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 describe the behavior.
- **Flow-oriented elements.** Information is transformed as it flows through a computer-based system. The system accepts input in a variety of forms, applies functions to transform it, and produces output in a variety of forms.

Elaboration – used elements

- **Some examples:**

SafeHome

Developing a High-Level Use-Case Diagram

The scene: A meeting room, continuing the requirements gathering meeting

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; and a facilitator.

The conversation:

Facilitator: We've spent a fair amount of time talking about *SafeHome* home security functionality. During the break I sketched a use case diagram to summarize the important scenarios that are part of this function. Take a look.

(All attendees look at Figure 5.2.)

Jamie: I'm just beginning to learn UML notation.[14] So the home security function is represented by the big box with the ovals inside it? And the ovals represent use cases that we've written in text?

Facilitator: Yep. And the stick figures represent actors—the people or things that interact with the system as described by the use case . . . oh, I use the labeled square to represent an actor that's not a person . . . in this case, sensors.

Doug: Is that legal in UML?

Facilitator: Legality isn't the issue. The point is to communicate information. I view the use of a humanlike stick figure for representing a device to be misleading. So I've adapted things a bit. I don't think it creates a problem.

Vinod: Okay, so we have use-case narratives for each of the ovals. Do we need to develop the more detailed template-based narratives I've read about?

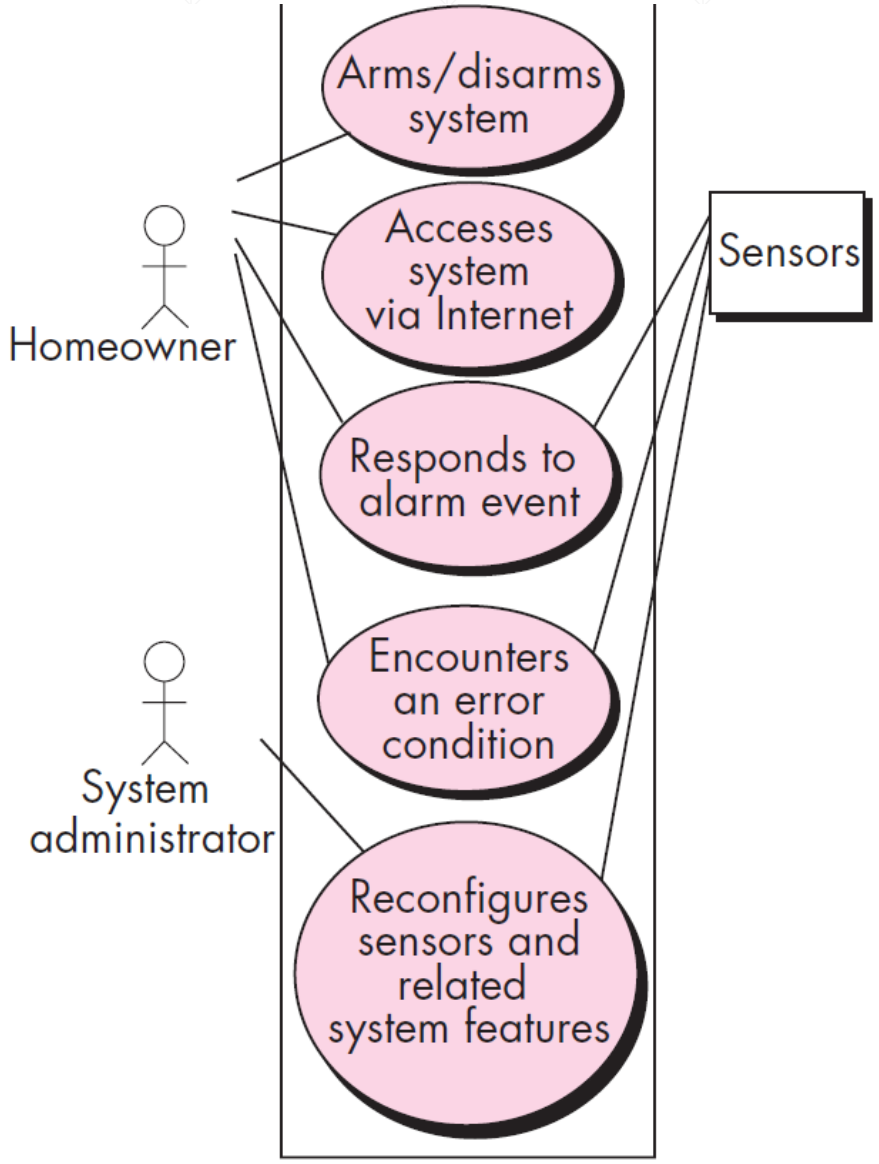
Facilitator: Probably, but that can wait until we've considered other *SafeHome* functions.

Marketing person: Wait, I've been looking at this diagram and all of a sudden I realize we missed something.

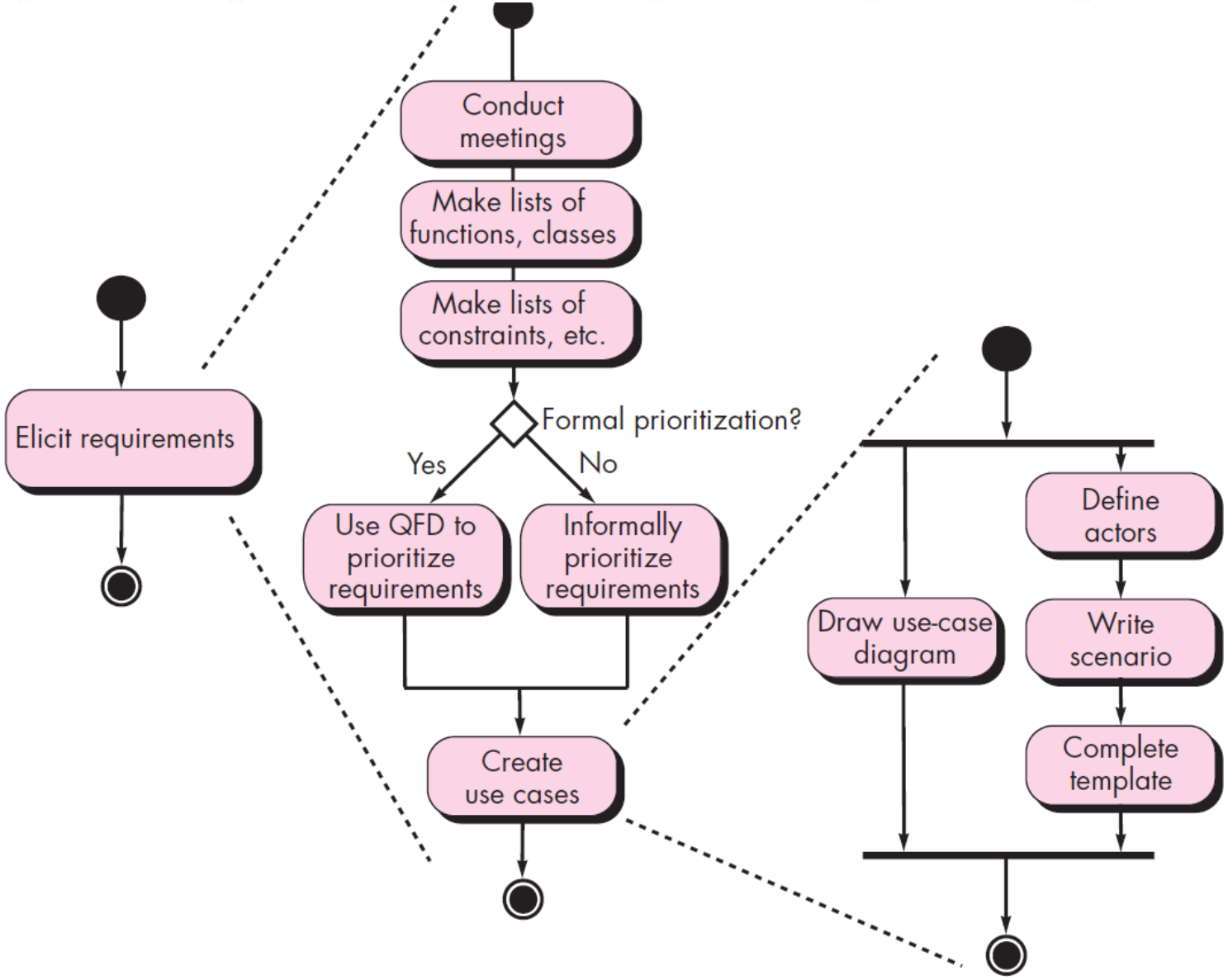
Facilitator: Oh really. Tell me what we've missed.

(The meeting continues.)

UML use case diagram for *SafeHome* home security function



UML activity diagrams for eliciting requirements



Source: Software Engineering Practitioner's Approach

Negotiation

The problem
Definition
Inception
Elicitation
Elaboration
Negotiation
Specification
Validation
Management

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 reconcile these conflicts through a process of **negotiation**. Customers, users, and other stakeholders are asked to **rank requirements and then discuss conflicts in priority**.

Negotiation

- In most cases, stakeholders are asked to balance functionality, performance, and other product or system characteristics against **cost and time-to-market**. The intent of this negotiation is to develop a project plan that meets stakeholder needs while at the same time reflecting the real-world constraints (e.g., time, people, budget) that have been placed on the software team.
- The best negotiations strive for a “win-win” result:
  - Stakeholders win by getting the system or product that satisfies the majority of their needs
  - You win by working to realistic and achievable budgets and deadlines.

Negotiation

INFO

The Art of Negotiation

Learning how to negotiate effectively can serve you well throughout your personal and technical life. The following guidelines are well worth considering:

1. *Recognize that it's not a competition.* To be successful, both parties have to feel they've won or achieved something. Both will have to compromise.
2. *Map out a strategy.* Decide what you'd like to achieve; what the other party wants to achieve, and how you'll go about making both happen.
3. *Listen actively.* Don't work on formulating your response while the other party is talking. Listen to her. It's likely you'll gain knowledge that will help you to better negotiate your position.
4. *Focus on the other party's interests.* Don't take hard positions if you want to avoid conflict.
5. *Don't let it get personal.* Focus on the problem that needs to be solved.
6. *Be creative.* Don't be afraid to think out of the box if you're at an impasse.
7. *Be ready to commit.* Once an agreement has been reached, don't waffle; commit to it and move on.

Source: Software Engineering Practitioner's Approach

Negotiation

Documentar os conflitos entre *stakeholders* incluindo:

- **Identificação dos conflitos** – Expor e descrever os conflitos existentes entre *stakeholders*
- **Análise dos conflitos** – Verificar a forma e como os conflitos ocorrem
- **Resolução de conflitos** – Identificar a forma de resolver os conflitos e negociar com os diferentes *stakeholders* uma forma de compatibilização
- **Aceitação da resolução de conflitos** – Aceitação por parte de todos os *stakeholders*

Specification

The problem
Definition
Inception
Elicitation
Elaboration
Negotiation
Specification
Validation
Management

Specification

- A Software Requirements Specification (SRS) 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.

Specification

- For large systems, a written document, combining natural language descriptions and graphical models may be the best approach. However, usage scenarios may be all that are required for smaller products or systems that reside within well-understood technical environments.
- Example of Software Requirement Specification (SRS):

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.

Source: Software Engineering Practitioner's Approach

ISO/IEC/IEEE 29148:2011 Systems and Software Engineering Life cycle processes — Requirements engineering

- International Standard
- It proposes several deliverables including the Software Requirements Specification document
- It suggests a lot of detail. Is this possible at the requirements gathering phase?
- Compliance with the current agile methodologies?
- Appropriate for large software products?

Adapt to your needs

- 9.5 Software requirements specification (SRS) document
  - 9.5.1 Purpose
  - 9.5.2 Scope
  - 9.5.3 Product perspective
    - 9.5.3.1 System interfaces
    - 9.5.3.2 User interfaces
    - 9.5.3.3 Hardware interfaces
    - 9.5.3.4 Software interfaces
    - 9.5.3.5 Communications interfaces
    - 9.5.3.6 Memory constraints
    - 9.5.3.7 Operations
    - 9.5.3.8 Site adaptation requirements
  - 9.5.4 Product functions
  - 9.5.5 User characteristics
  - 9.5.6 Limitations
  - 9.5.7 Assumptions and dependencies
  - 9.5.8 Apportioning of requirements
  - 9.5.9 Specific requirements
  - 9.5.10 External interfaces
  - 9.5.11 Functions
  - 9.5.12 Usability requirements
  - 9.5.13 Performance requirements
  - 9.5.14 Logical database requirements
  - 9.5.15 Design constraints
  - 9.5.16 Standards compliance
  - 9.5.17 Software system attributes
  - 9.5.18 Verification
  - 9.5.19 Supporting information

Source: Software Engineering Practitioner's Approach

1. **Introduction**
   - 1.1 Purpose
   - 1.2 Document conventions
   - 1.3 Project Scope
   - 1.4 References
2. **Overall Description**
   - 2.1 Product Perspective
   - 2.2 User Classes and Characteristics
   - 2.3 Operating Environment
   - 2.4 Design and Implementation Constraints
   - 2.5 Assumptions and Dependencies
3. **System Features**
   - 3.x System feature x
     - 3.x.1 Description
     - 3.x.2 Functions requirements
4. **External Interface Requirements**
   - 4.1 User Interfaces
   - 4.2 Hardware Interfaces
   - 4.3 Software Interfaces
   - 4.4 Communications Interfaces
5. **Quality Attributes**
   - 5.1 Usability
   - 5.2 Performance
   - 5.3 Security
   - 5.4 Safety
   - 5.x [others]
6. **Internationalization and localization requirements**
7. **Other Requirements**

Appendix A: Glossary

Appendix B: Analysis models

Source: Software Requirements (Third Edition) by Joy Beatty & Karl Wiegers

SRS

1. **Introduction**

1.1 Purpose

1.2 Document conventions

1.3 Project Scope

1.4 References

2 **Overall Description**

2.1 Product Perspective

2.2 User Classes and Characteristics

2.3 Operating Environment

2.4 Design and Implementation

2.5 Assumptions and Depend

3. **System Features**

3.x System feature x

3.x.1 Description

3.x.2 Functions requirements

Ex. Th
require
ir

Ex.
orde
cam

3.1 Ex. Order Meals

3.1.1 Description and Priority

A cafeteria Patron whose identity has been verified may order meals either to be delivered to a specified company location or to be picked up in the cafeteria. A Patron may cancel or change a meal order if it has not yet been prepared. Priority = High.

3.1.2 **Functional Requirements**

Order.Place: The system shall let a Patron who is logged into the Cafeteria Ordering System place an order for one or more meals.

Order.Place.Register: The system shall confirm that the Patron is registered for payroll deduction to place an order.

Order.Place.Date:The system shall prompt the Patron for the meal date (see BR-8).

Order.Place.Date.Cutoff: If the meal date is the current date and the current time is after the order cutoff time, the system shall inform the patron that it's too late to place an order for today. The Patron may either change the meal date or cancel the order.

Source: Software Requirements (Third Edition) by Joy Beatty & Karl Wiegers

1. **Introduction**

Ex. The Web pages shall permit complete navigation and fo
to

SI-1: Cafeteria Inventory System

The Cafeteria Ordering System shall send an e-mail message to the Patron to confirm acceptance of an order, price, and delivery instructions.

The system shall accommodate 400 users during the

All network transactions that involve financial information or personally identifiable information shall be encrypted per BR-33.

the menu for the current date.

3.x System feature x

3.x.1 Description

3.x.2 Functions requirements

4. External Interface Requirements

4.1 User Interfaces

4.2 Hardware Interfaces

4.3 Software Interfaces

4 Communications Interfaces

Quality Attributes

5.1 Usability

2 Performance

curity

Safety

[others]

Internationalization and localization requirements

7. Other Requirements

Appendix A: Glossary

Appendix B: Analysis models

Validating

The problem
Definition
Inception
Elicitation
Elaboration
Negotiation
Specification
Validation
Management

Validating

- As each element of the requirements model is created, it is examined for inconsistency, omissions, and ambiguity.
- The requirements represented by the model are prioritized by the stakeholders and grouped within requirements packages that will be implemented as software increments.

"How to torture a Business Analyst - #1"

Validating – What questions should I ask?

- Is each requirement consistent with the overall objectives for the system/product?
- Have all requirements been specified at the proper level of abstraction 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?
- Do any requirements conflict with other requirements?
- Is each requirement testable, once implemented?
- Etc.

Management

The problem
Definition
Inception
Elicitation
Elaboration
Negotiation
Specification
Validation
Management

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.

Requirements Management

- Each requirement is assigned a unique identifier
- The requirements are then placed into one or more traceability tables
- These tables may be stored in a database(or excel) that relate features, sources, dependencies, subsystems, and interfaces to the requirements
- A requirements traceability table is also placed at the end of the software requirements specification

Sources

- Software Engineering – A Practitioner's Approach(seventh edition) by Roger S. Pressman
- Software Requirements (Third Edition) by Joy Beatty & Karl Wiegers