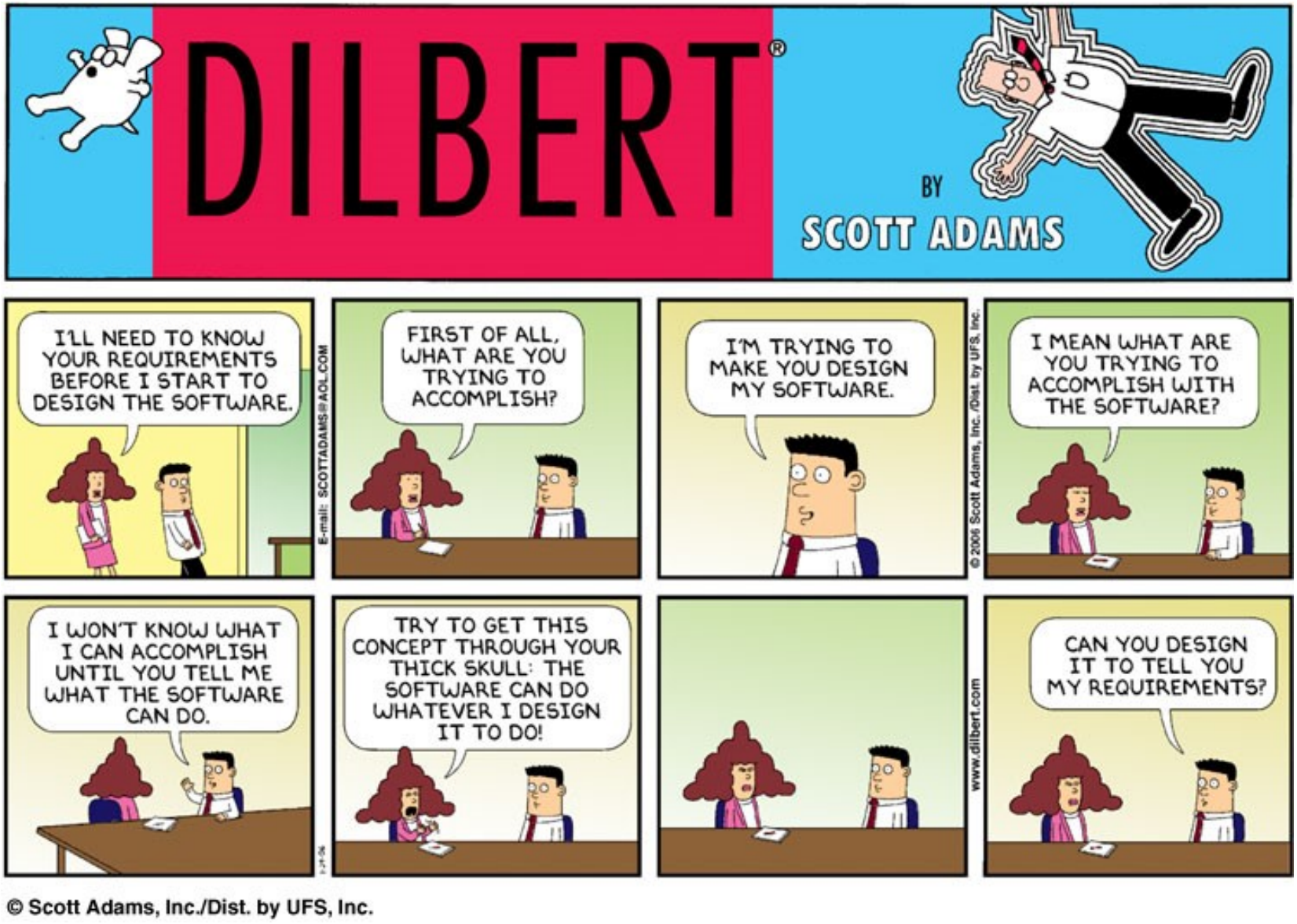
Raise Your Hands

If You Feel You Have Learned something last week (Lecture, Assignment)

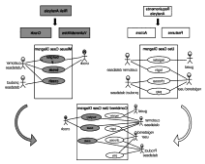
If You Feel You Have enjoyed my absence

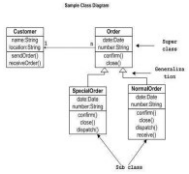
If You Feel You feel like having a chocolate

- Jayden Khakurel
- Jayden.khakurel@csulb.edu



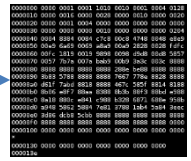
http://blogs.warwick.ac.uk/images/steverumsby/2006/01/30/dilbert20060121046729.jpg





What do we have before that?

```
if (hour < 18) {
   greeting = "Good day";
} else {
   greeting = "Good evening";
}
```



Requirements

Analysis

Design

Capturing the purpose of a system

Understanding Requirements

Some of the slides are designed and adapted from slides provided by Software Engineering: A Practitioner's Approach, 8e

What are requirements?

- An expression of the ideas to be embodied in the system or application under development

- A statement about the proposed system that all stakeholders agree must be made true in order for the customer's problem to be adequately solved

  - Short and concise piece of information
  - Says something about the system
  - All the stakeholders have agreed that it is valid
  - It helps solve the customer's problem

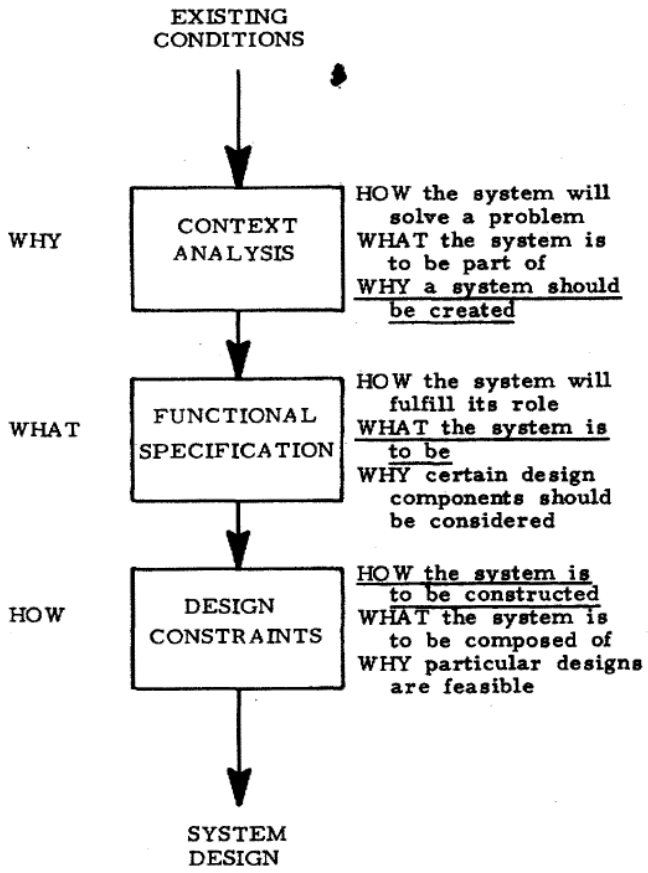
Understanding requirements

According to IEEE 830-1993

- A **requirement** is defined as:
  - A condition or capability needed by a user to solve a problem or achieve an objective
  - A condition or a capability that must be met or possessed by a system … to satisfy a contract, standard, specification, or other formally imposed document …

Understanding Requirements

- Good requirements will answer all the important questions.
- Think about the **why, what, how, where, when, who.**



- **How will you use this feature?**
  - **What is the end result of doing this?**
    - **When will this feature be used?**
  - **Is there any other way to accomplish this?**
    - **Who will use this feature?**
  - **Where would the results be visible?**

Ross et al.(1977), Structured Analysis for Requirements Definition

Understanding Requirements Engineering

Some of the slides are designed and adapted from slides provided by Software Engineering: A Practitioner's Approach, 8e

Requirements engineering

- **Requirements engineering is**
  - **an** interdisciplinary function that mediates between the **domains of the acquirer** and **supplier** to **establish and maintain the requirements** to be **met by the system, software or service of interest**. (ISO/IEC/IEEE 29148)
  - **Broad spectrum** of **tasks and techniques** that **lead to an understanding of requirements** (Pressman et.al)
- The **result of requirements engineering** is a hierarchy of requirements that:
  - enables an agreed understanding between stakeholders (e.g., acquirers, users, customers, operators, suppliers)
  - is validated against real-world needs, can be implemented
  - provides a basis of verifying designs and accepting solutions.

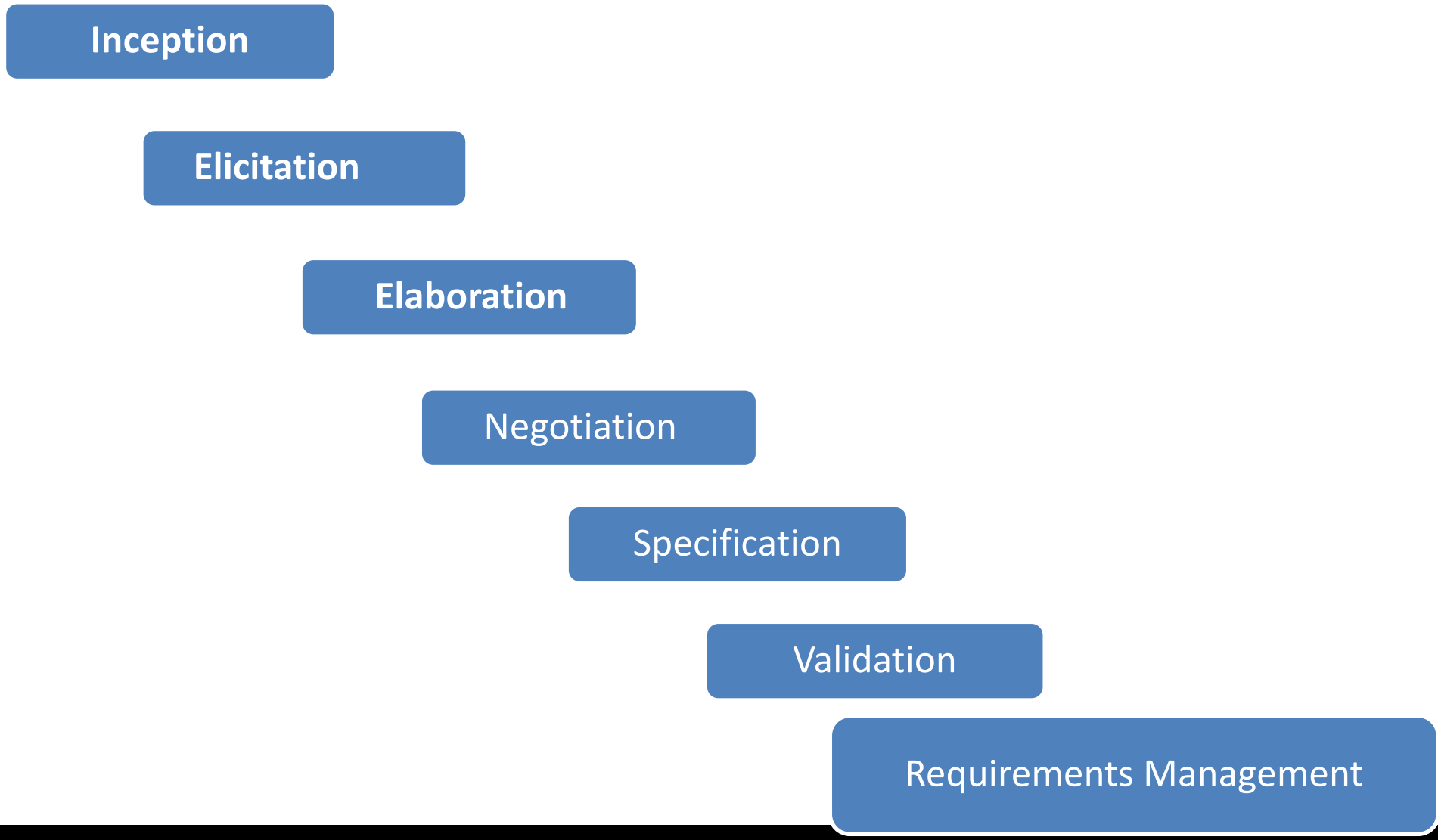
Source: ISO/IEC/IEEE 29148 http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6146379

Requirements engineering

"the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behavior, and to their evolution over time and across software families." (Zave, 1997)

(Classification of Research Efforts in Requirements Engineering. ACM Computing Surveys (1997)) http://dl.acm.org/citation.cfm?id=267581

RE encompasses distinct tasks



Inception

Start the process with conversation

With whom? Stakeholders from the business community(e.g. business managers, marketing people, product managers, potential users)

About what? (business need, market opportunity, great idea, ...), business case, feasibility study, system scope, risks, etc.

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.

establish a basic **understanding of the problem**, the **people who want a solution**, the **nature of the solution** that is desired

Elicitation

- **Communication** is the Key
- is to establish business goals.
- job is to engage stakeholders and to encourage them to share their goals honestly.
- Once the goals have been captured
- a **prioritization mechanism should be established**, and a **design rationale , for a potential architecture** (that meets stakeholder goals) can be created.

However.....

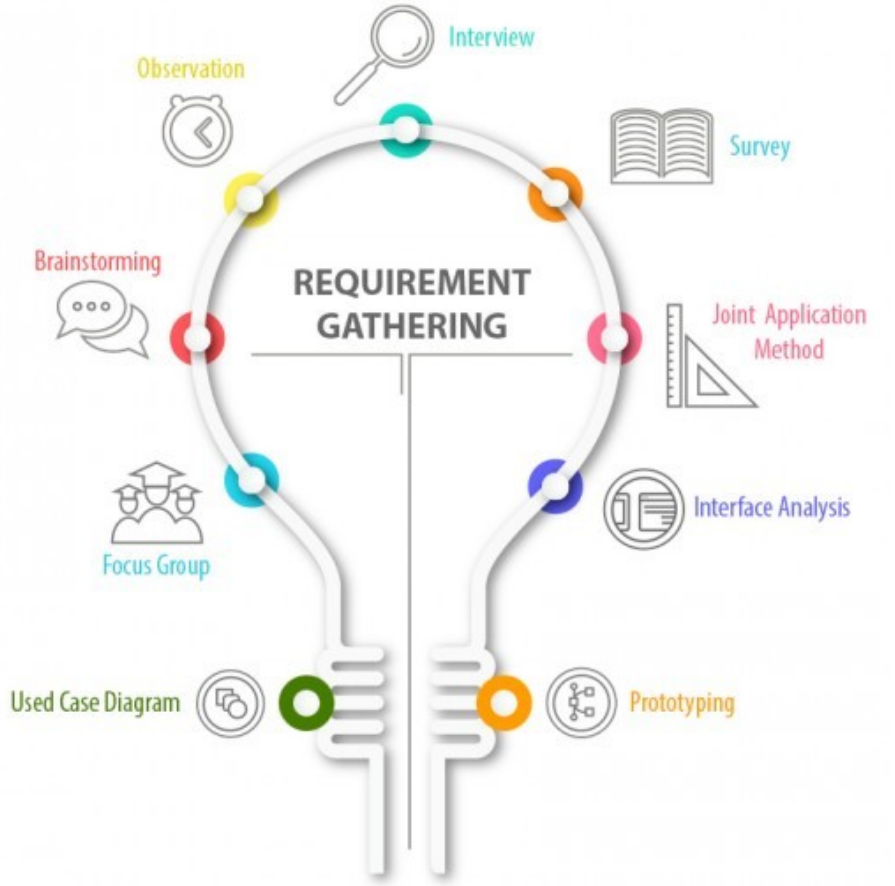
Elicitation

- **WHAT IS THE SOLUTION?**
- approach the requirements-gathering activity in an organized manner.

PLAN

COMMUNICATE

UNDERSTAND CUSTOMER



https://nmgtechnologies.com/requirement-gathering-solve-biggest-problems-consulting.html

Elaboration

- What happens?
  - Information obtained during inception and elicitation from customer is explained and refined
- Focuses on
  - Developing a refined requirements model
- Such as
  - scenario based methods
  - Class based methods
  - Behavior, patterns

,

- identifies various aspects of software function, behavior and intention

Negotiation

- What happens?
  - customers and users to ask for more than can be achieved, given limited business resources.
  - different customers or users to propose conflicting requirements,
  - Upcoming version is “essential for our special needs.”
- Leads to conflicts

Negotiation

- How?
  - 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
  - each party achieves some measure of satisfaction.

There should be no winner and no loser in an effective negotiation. Both sides win, because a "deal" that both can live with is solidified.

Specification

- Has different meaning?
  - 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**.
  - Sometimes its better to flexible when a specification is to be developed

    For large systems, a written document, combining natural language descriptions and graphical models may be the best approach.

    For small products or system,
    usage scenarios may be all that are required if that reside within well-understood technical environments.

Validation

- Work products as a consequence of requirements engineering are assessed
  - examine the specification to ensure that all software requirements have been stated unambiguously
  - NO inconsistencies, omissions, and errors have been detected and corrected;
  - work products conform to the standards established for the process, the project, and the product.

A key concern during requirements validation is consistency. Use the analysis model to ensure that requirements have been consistently stated.

Validation

- Who does the validation
  - Review team (e.g. software engineers, customers, users, and other stakeholders)
- What do they examine
  - specification looking 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 requirements,
  - Unrealistic (unachievable) requirements.

https://nmgtechnologies.com/requirement-gathering-solve-biggest-problems-consulting.html

Validation

- Who does it?
  - Review team (e.g. software engineers, customers, users, and other stakeholders)
- What do they examine
  - specification looking 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 requirements,
  - Unrealistic (unachievable) requirements.

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?

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.

Establishing the Groundwork

Some of the slides are designed and adapted from slides provided by Software Engineering: A Practitioner's Approach, 8e

Steps to establish the Ground work

- Identifying stakeholders
- Recognizing multiple view points
- Working towards collaboration
- Asking the first question
- Non Functional requirements
- Traceability
- Requirements Management

Identifying stakeholders

- Sommerville and Sawyer [Som97] define a stakeholder as “anyone who benefits in a direct or indirect way from the system which is being developed.”
- 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, you should create a list 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?”

Recognizing multiple viewpoints

- Multiple stakeholders exist, the requirements of the system will be explored from many different points of view.
- Marketing, support engineers, end users,
- As information from multiple viewpoints is collected, emerging requirements may be inconsistent or may conflict with one another.
- categorize all stakeholder information (including inconsistent and conflicting requirements) in a way that will allow decision makers to choose an internally consistent set of requirements for the system.

Working toward collaboration

- Multiple stakeholders meaning different opinions about the proper set of requirements
- job of a requirements engineer is to identify areas of commonality (i.e., requirements on which all stakeholders agree)
- areas of conflict or inconsistency (i.e., requirements that are desired by one stakeholder but conflict with the needs of another stakeholder).

Asking the first question

- Questions asked at the inception of the project should be “context free” [Gau89].
- overall project goals and benefits.

Who is behind the request for this work? **Who will use the solution?**

- understanding of the problem and allow the customer to voice his or her perceptions about a solution

What problem(s) will this solution address?

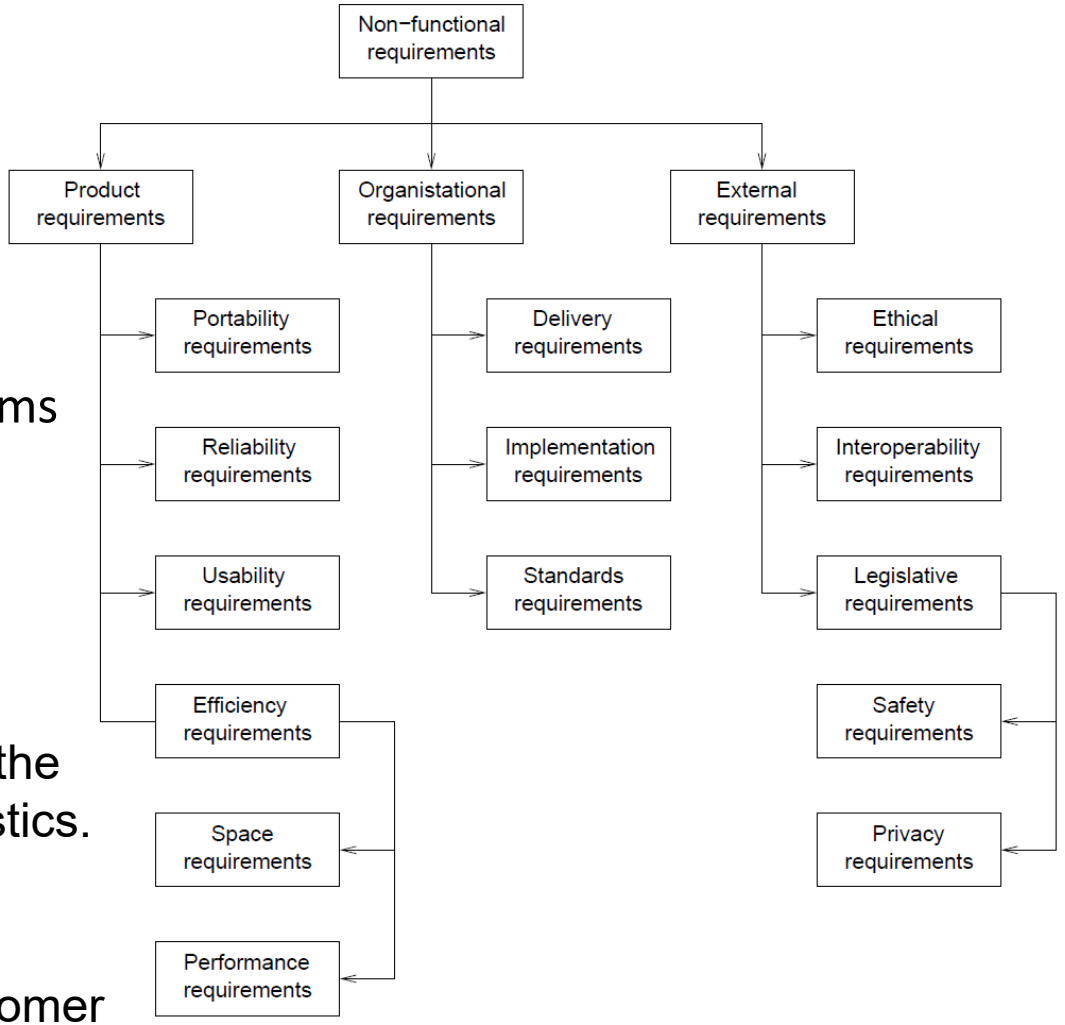
- the effectiveness of the communication activity itself.

Are my questions relevant to the problem that you have?

Should I be asking you anything else?

Non Functional requirements

- Can be described as
  - Quality attribute
  - performance attribute
  - Security attribute
  - General constraints on a systems



often not easy for stakeholders to articulate

may not be useful or usable without the necessary non-functional characteristics.

Quality function deployment (QFD) attempts to translate unspoken customer needs

Ref: http://www.inf.ed.ac.uk/teaching/courses/cs2/LectureNotes/CS2Ah/SoftEng/se02.pdf

Traceability

- is a software engineering term that refers to documented links between software engineering work products (e.g., requirements and test cases).
  - Traceability matrix
    - allows a requirements engineer to represent the relationship between requirements and other software engineering work products.

Unique Req ID	Requirement description	Source /Requestor	Org /Dept	Business Justification/Need	WBS Deliverable	Test Strategy	UAT Responsibility	Status	Active/ Inactive Flag	Comments
1	Change the table component on the dashboard to a graph.	Ella Allen	Sales	Better representaion of the data and improved readability	Task 1.1 Task 4.7	Use cases to be developed.	Follow the test steps as defined in use cases and report any defects.	Done	Active	Jan 5:- Testing started. Jan 8:- Defected reported. Jan 9: Defect fixed Jan 10: UAT Continued
2	Add a drop down list for the regions	Tonya Harper	Sales	Will enable Area managers to understand their market more accurately	Task 1.2	Load testing to be done.	Load runner to be used to simulate a load and the regions will be verified for accurate representation	In Progress	Active	Make sure US territories hawaii and peurto rico are included in a separate regional unit.
3	Create a new category hierarchy to sorting the result set	Sammy Butler	Pricing	Will help the pricing department by automating the selection of categorized data.	Task 1.3	Assigned business users to perform unit testing as well as UAT	Check the categories in the base tables in the EDW.	Hold	Cancelled	It was determined that Pricing requirements were out of scope for this phase
4	...	...	...	...	...	...	...	...	...	...
5	...	...	...	...	...	...	...	...	...	...
6	...	...	...	...	...	...	...	...	...	...
7	...	...	...	...	...	...	...	...	...	...
8	...	...	...	...	...	...	...	...	...	...

https://project-management.com/requirements-traceability-matrix-rtm/

Eliciting Requirements

Some of the slides are designed and adapted from slides provided by Software Engineering: A Practitioner's Approach, 8e

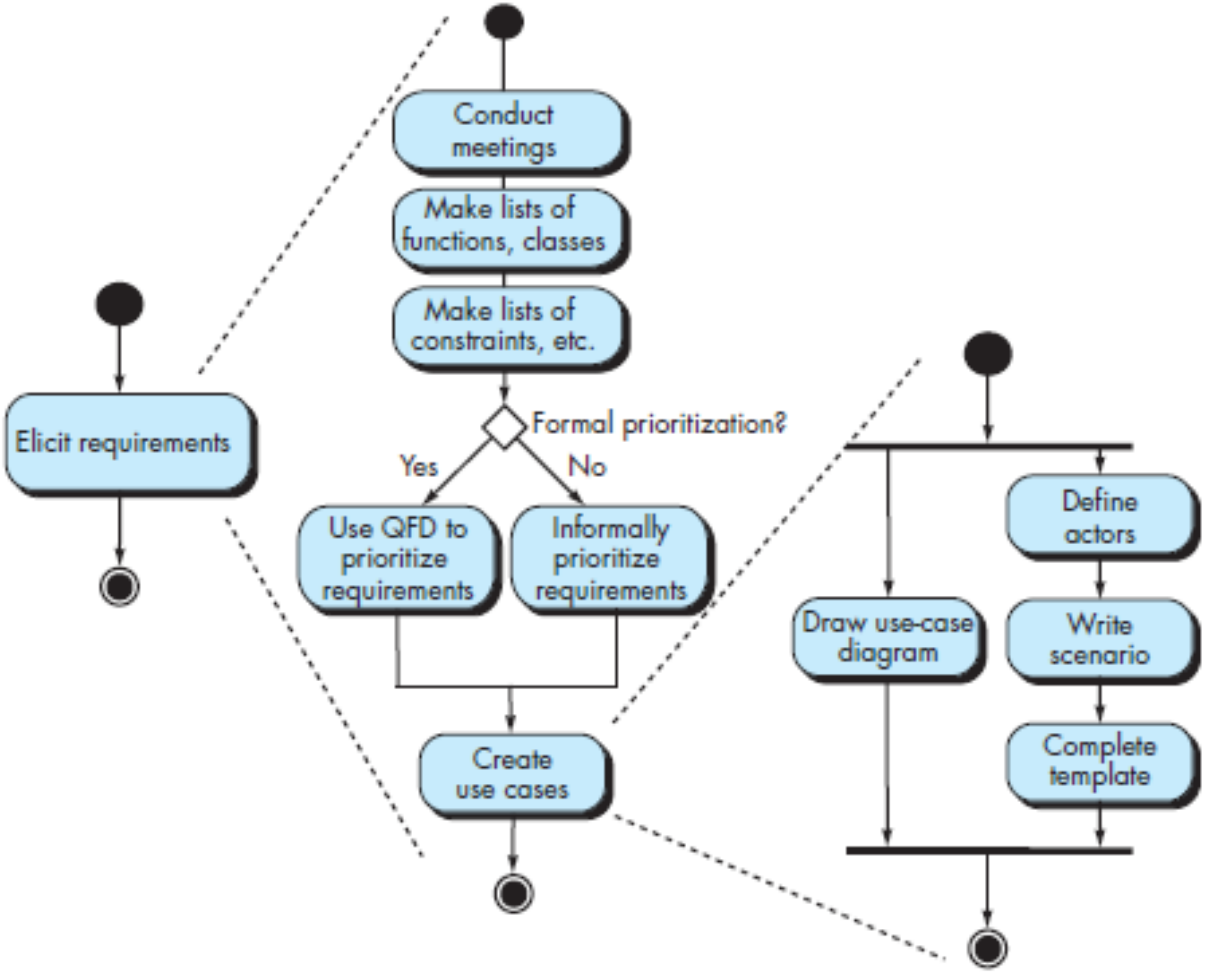
Eliciting Requirements

- meetings are conducted and attended by both software engineers and customers
- rules for preparation and participation are established
  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

Collaborative requirements gathering

- What are the basic guidelines for conducting a collaborative Requirements gathering meeting?
  - Meetings (either real or virtual) 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.

Eliciting requirements



Elicitation work products

- 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.
- 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.

https://project-management.com/requirements-traceability-matrix-rtm/

Building Analysis Model

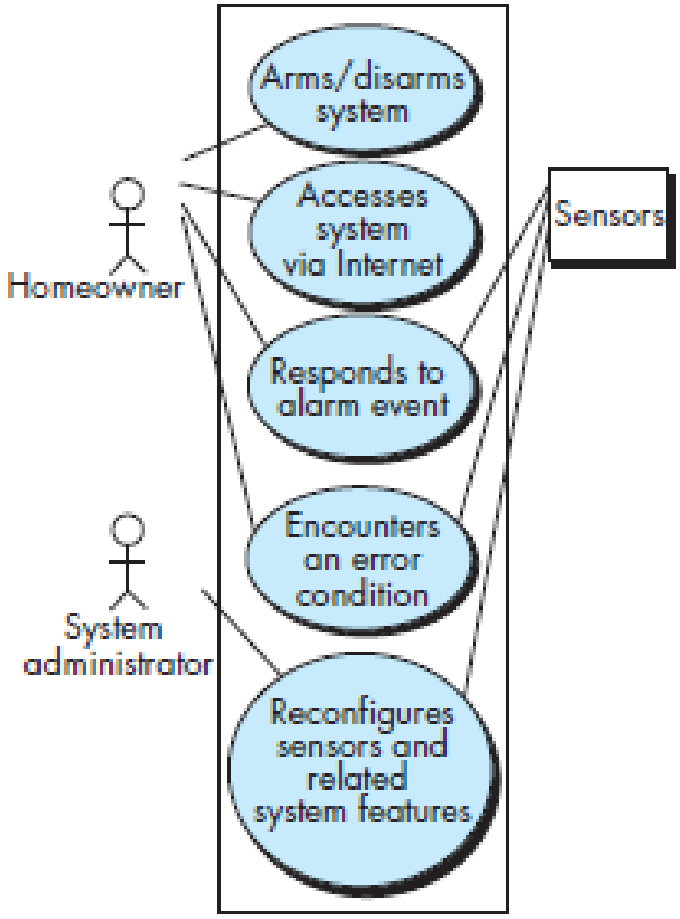
- Elements of the analysis model
  - Scenario-based elements
    - Functional—processing narratives for software functions
    - Use-case—descriptions of the interaction between an “actor” and the system
  - Class-based elements
    - Implied by scenarios
  - Behavioral elements
    - State diagram
  - Flow-oriented elements
    - Data flow diagram

Developing 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?

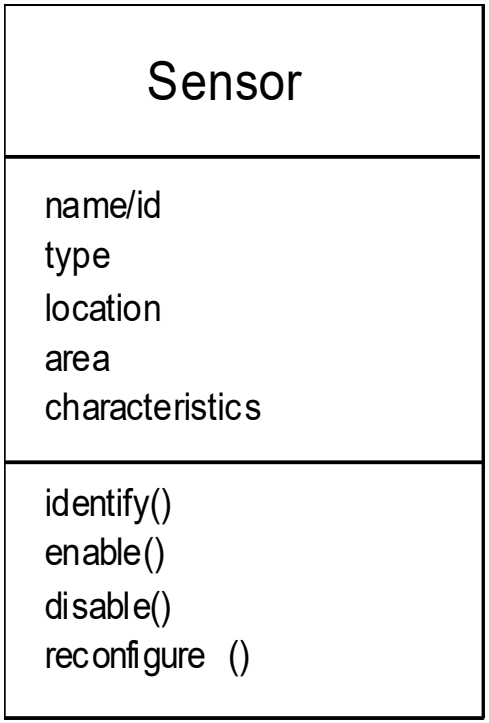
https://project-management.com/requirements-traceability-matrix-rtm/

Use-case diagram



Class diagram

From the *SafeHome* system …



Analysis Patterns

Pattern name: A descriptor that captures the essence of the pattern.

Intent: Describes what the pattern accomplishes or represents

Motivation: A scenario that illustrates how the pattern can be used to address the problem.

Forces and context: A description of external issues (forces) that can affect how the pattern is used and also the external issues that will be resolved when the pattern is applied.

Solution: A description of how the pattern is applied to solve the problem with an emphasis on structural and behavioral issues.

Consequences: Addresses what happens when the pattern is applied and what trade-offs exist during its application.

Design: Discusses how the analysis pattern can be achieved through the use of known design patterns.

Known uses: Examples of uses within actual systems.

Related patterns: One or more analysis patterns that are related to the named pattern because (1) it is commonly used with the named pattern; (2) it is structurally similar to the named pattern; (3) it is a variation of the named pattern.

Negotiating Requirements

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

- 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?
- Do any requirements conflict with other requirements?

Validating 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.
- Have requirements patterns been used to simplify the requirements model. Have all patterns been properly validated? Are all patterns consistent with customer requirements?

- Read the given materials in class
- Brief summary (more information on beach board)