CT70A2000
Requirements Engineering

Shola Oyedeji

LUT University

LECTURE 2

Requirements Elicitation

Requirements Analysis

Requirements Prioritization

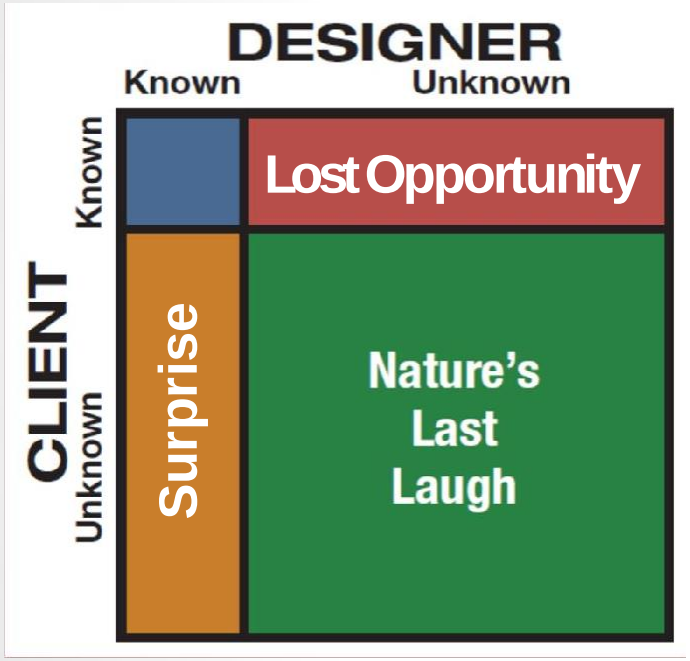
I want a software that allows me to record lecture and transcribe them into lecture note

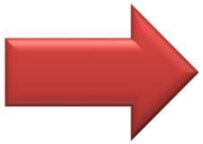
WHY REQUIREMENTS DEVELOPMENT IS DIFFICULT?

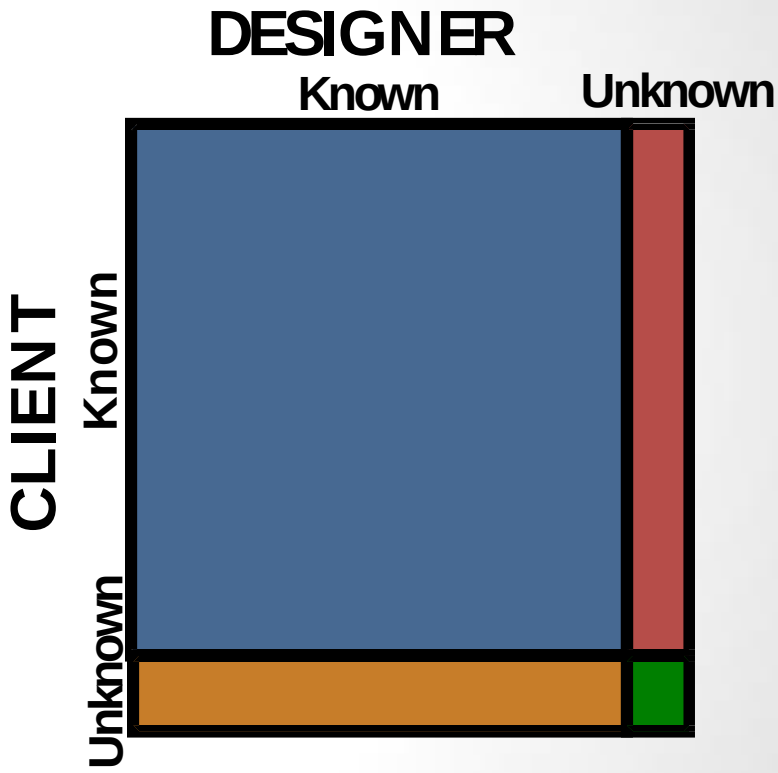
- **Customers and users often can't specify their requirements alone.**
  - They often don't understand how software design and development works.
- **Software developers can't specify requirements alone for the customers.**
  - They often do not understand the problems and needs of customers and users well enough.

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

WHY TO ELICITATE AND DEVELOP REQUIREMENTS?







Gause, Donald C., and Brian Lawrence. "User-driven design." *Software Testing & Quality Engineering* 1.1 (1999): 22-28.
http://stareast.techwelldev.com/sites/default/files/articles/Smzr1XDD2659filelistfilename1_0.pdf

TYPICAL REQUIREMENTS DEVELOPMENT DIFFICULTIES

- Differing and sometimes conflicting needs among different types of users
- Unstated or assumed requirements on the part of stakeholders
- Gaining access to knowledgeable stakeholders
- An inability to envision new or different ways to use software
- Uncertainty about how to adapt to changing business needs
- Having a large numer of highly interrelated requirements
- Having limited time to elicit requirements from busy stakeholders
- Overcoming resistance to change

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

REQUIREMENTS BILL OF RIGHTS FOR SOFTWARE CUSTOMERS

You have the right to:

1. Expect analysts to speak your language.
2. Expect analysts to learn about your business and your objectives for the system.
3. Expect analysts to structure the information you present during requirements elicitation into a written software requirements specification.
4. Have analysts explain all work products created from the requirements process.
5. Expect analysts and developers to treat you with respect and to maintain a collaborative and professional attitude throughout your interactions.
6. Have analysts and developers provide ideas and alternatives both for your requirements and for implementation of the product.
7. Describe characteristics of the product that will make it easy and enjoyable to use.
8. Be given opportunities to adjust your requirements to permit reuse of existing software components.
9. Receive good-faith estimates of the costs, impacts, and tradeoffs when you request a change in the requirements.
10. Receive a system that meets your functional and quality needs, to the extent that those needs have been communicated to the developers and agreed upon.

K. E. Wiegers, *Software Requirements*. Microsoft Press, 2003.

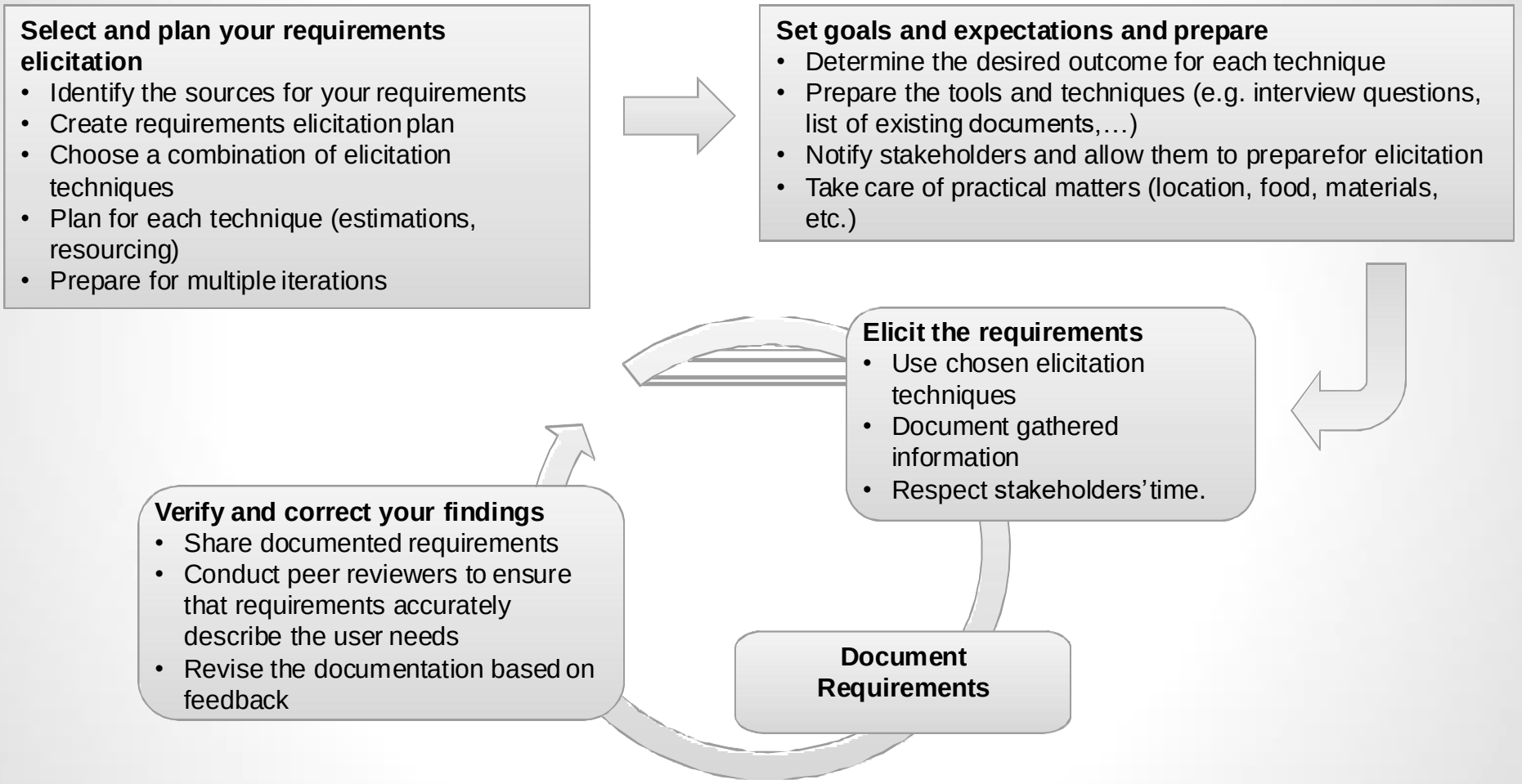
REQUIREMENTS BILL OF RESPONSIBILITIES FOR SOFTWARE CUSTOMERS

You have the responsibility to:

1. Educate analysts and developers about your business and define business jargon.
2. Spend the time that it takes to provide requirements, clarify them, and iteratively flesh them out.
3. Be specific and precise when providing input about the system's requirements.
4. Make timely decisions about requirements when requested to do so.
5. Respect a developer's assessment of the cost and feasibility of requirements.
6. In collaboration with the developers, set priorities for functional requirements, system features, or use cases.
7. Review requirements documents and evaluate prototypes
8. Communicate changes to the requirements as soon as you know about them.
9. Follow the development organization's process for requesting requirements changes.
10. Respect the processes the analysts use for requirements engineering.

K. E. Wiegers, *Software Requirements*. Microsoft Press, 2003.

HOW TO ELICIT REQUIREMENTS?



Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

WHERE DO REQUIREMENTS COME FROM?

- Interviews and discussions with potential users
- Facilitated workshops, Focus groups
- Prototypes
- Documents that describe current or competing products
- System requirements specifications
- Problem reports and enhancement requests for a current system
- Marketing surveys and user questionnaires
- Observing users at work
- Scenario analysis of user tasks
- Events and responses

K. E. Wiegers, *Software Requirements*. Microsoft Press, 2003.

Roles that promote user viewpoint

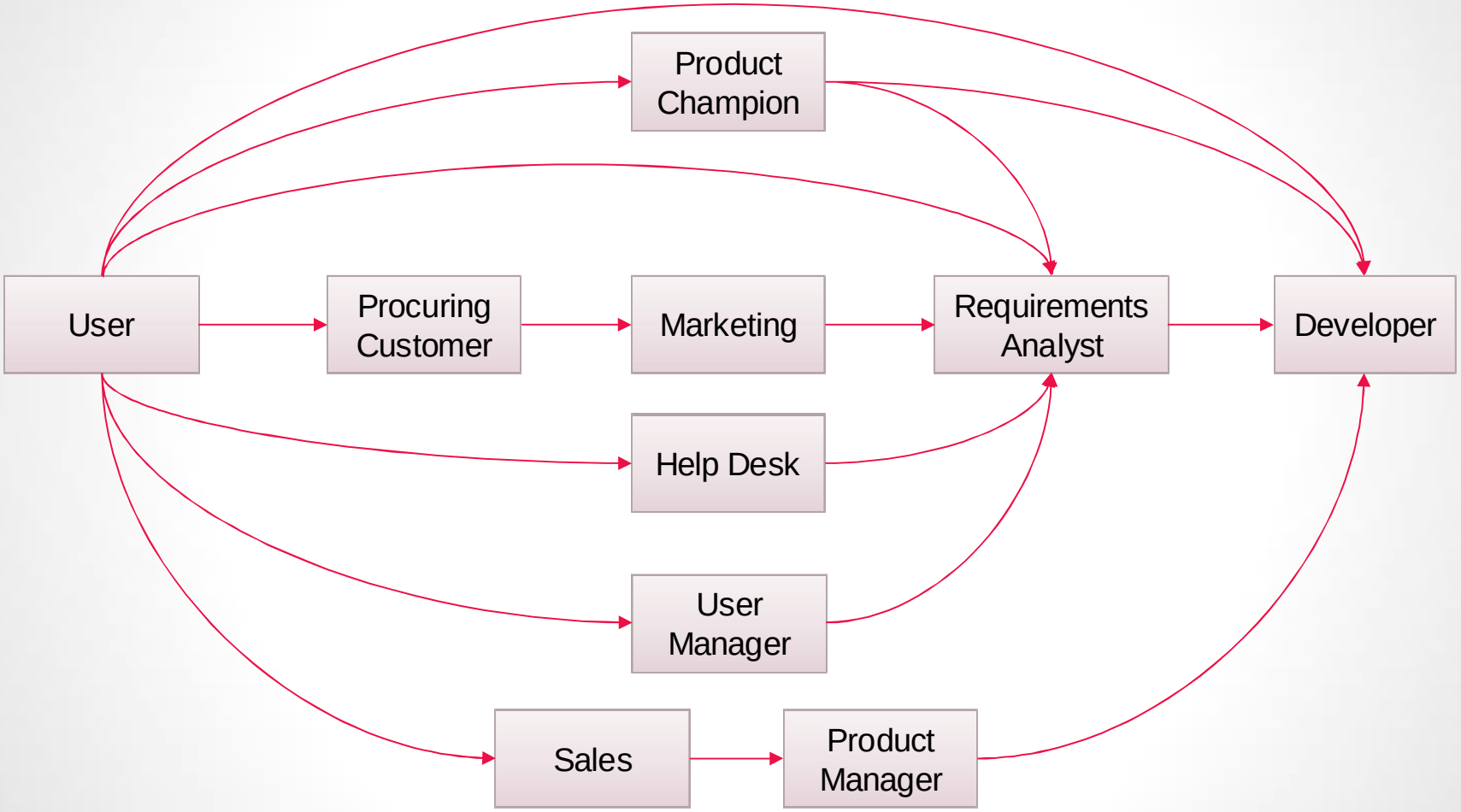
- **Surrogate** is someone taking the place of actual user.
- **Product Champion** is a key member of user community that helps to provide the requirements.
  - Each product champion serves the primary interface between members of a single user class and the project's requirements analyst.
  - The best product champions have a clear vision of the new system and are enthuastic about it because they see how it will benefit them and their peers.
  - They should be effective communicators who are respected by their colleagues.
  - They need a thorough understanding of the application domain and tye system's operating environment.

Possible Product Champion Activities

Category	Activities
Planning	• Refine the scope and limitations of the product • Define interfaces to other systems • Evaluate impact of new system on business operations • Define a transition path from current applications
Requirements	• Collect requirements from other users • Develop usage scenarios and use cases • Resolve conflicts between proposed requirements • Define implementation priorities • Specify quality and performance requirements • Evaluate user interface prototypes
Validation and Verification	• Inspect requirements documents • Define user aceptance criteria • Develop test cases from usage scenarios • Provide test data sets • Perform beta testing
User Aids	• Write portions of user manuals and help text • Prepare training materials for tutorials • Present product demonstrations to peers
Change Management	• Evaluate and prioritize defect corrections • Evaluate and prioritize enhancement requests • Evaluate the impact of proposed requirements changes on users and business processes • Participate in making change decisions

K. E. Wiegers, *Software Requirements* (Ch 6). Microsoft Press, 2003.

HOW USER NEEDS ARE COMMUNICATED TO THE DEVELOPERS



K. E. Wiegers, *Software Requirements* (Ch 6). Microsoft Press, 2003.

Tools and techniques to elicit requirements

When you need to:	Then create:
Identify sources of requirements	A Requirements Source List
Identify product stakeholders	Stakeholder Categories
Describe stakeholders' needs and succss criteria	Stakeholder Profiles
Review elicitation techniques	Identified Combinations of Elicitation Techniques: Interviews, Exploratory Prototypes, Facilitated Workshops, Focus Groups, User Task Analysis, Observation, Existing Documentation Study
Plan an elicitation approach	A Stakeholder Elicitation Plan

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

Requirements Source List

What is it?

Inventory of the people, specific documents and external information sources that you will elicit requirements from.

Why do it?

To identify potential documentation sources of requirements and allow analysts to elicit, review, document, and verify requirements information with stakeholders.

How to do it?

1. **Identify the relevant stakeholders that you should elicit requirements from.**
2. **Identify any documentation that you can use as a source of requirements information.** For example:
   - Existing and interfacing systems documentation
   - Change requests, software defect lists, customer complaint logs, and issues lists
   - User guides, training materials, and work procedures guidelines
   - Help desk documentation
3. **Identify external sources of information.** For example:
   - Departments or service companies that provide market survey data and industry analysis
   - Descriptions and reviews of competitive software products and product materials
   - Sales, marketing and communication materials
   - Regulations, guidelines, and laws from governmental agencies and regulatory bodies

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

Stakholder Categories

a.k.a. Stakeholder Classes, Stakeholder Statement

What are they?

Structured arrangements of groups or individuals who have a vested interest in the software product you are developing.

Why do it?

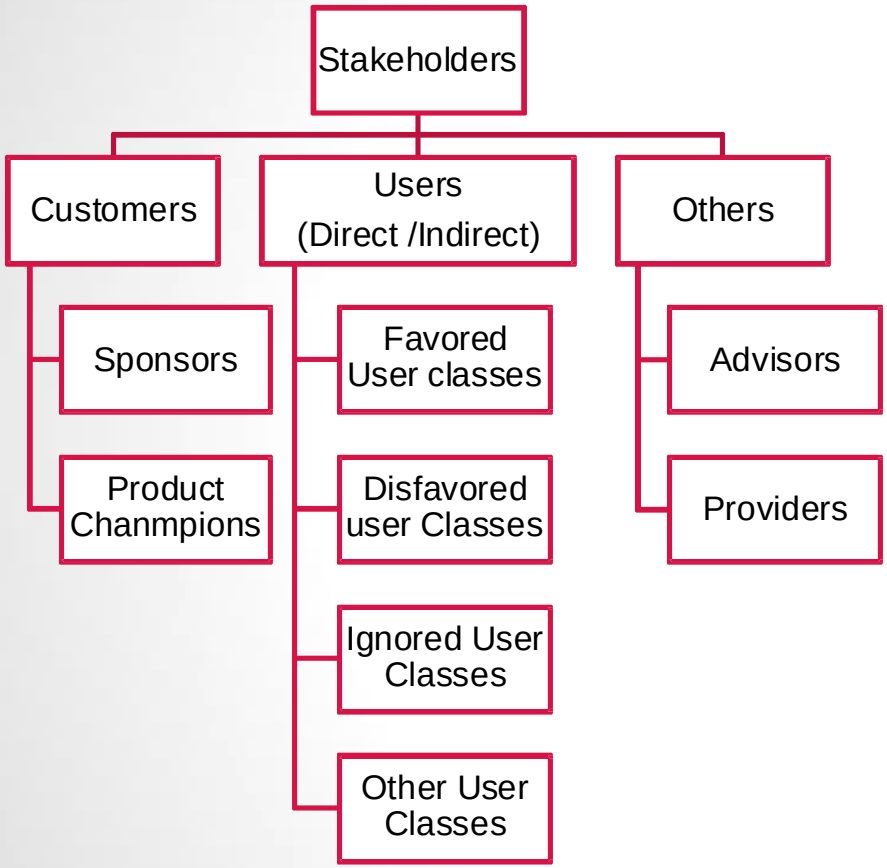
To understand who has an interest in or influence on the project, who will use the software and its outputs, and who the product will affect in some way.

How to do it?

1. **Identify stakeholders as either customers, users, or other stakeholders.**
2. **Review the list of stakeholder categories with the project stakeholders to ensure the list is complete and accurate.**
3. **Revise the list as needed and share it with the entire team**

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

CATEGORIZING STAKEHOLDERS



- **Customers** are responsible for accepting or paying the product
- **Sponsors** authorize or legitimize the product development effort by contracting or paying for the project
- **Product Champions** ensure that the software meets the needs of multiple user communities. They identify who should participate in requirements development and ensure that the right requirements are gathered.
- **Direct users** are the parties that directly interact with the software
- **Indirect users** do not directly interact with the system but may be affected by it.
- **Advisors** have relevant information about the product (subject matter experts)
- **Providers** design and produce the software.

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger
K. E. Wiegers, *Software Requirements* (Ch 6). Microsoft Press, 2003.

Example: Stakeholder Categories

Customers		Users		Other Stakeholders	
Sponsor	Product Champion	Direct Users	Indirect Users	Advisors	Providers
• CEO	• Office manager • …	• Office manager • Estimator • Scheduler • Inventory supply managers • …	• Credit card authorizers • Marketing manager • Advertising staff • …	• Tax accountant • Commercial real estate agents • …	• Project manager • Analyst • Developers

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

Stakeholder Profiles

a.k.a. Customer Profile, User Profile, Stakeholder Analysis

What are they? Description that characterizes each stakeholder and explains his or her relationship to the project.

Why do it? To understand the interests, concerns, and product success criteria for each of the system's' stakeholders, to uncover potential sources of requirements conflict among stakeholders, and to highlight requirements topics that may need additional time and attention. Stakeholder profiles can also reveal potential obstacles for successful product implementation and help you define how much involvement each stakeholder should have in requirements elicitation.

How to do it?

1. **Write a brief profile for each stakeholder. Describe his or her:**
   - *Role:* List the stakeholder category that the stakeholder belongs to.
   - *Responsibilities:* Briefly describe each stakeholder's role as it relates to the project.
   - *Interests:* List the stakeholder's needs, wants, and expectations for the product.
   - *Success criteria:* Describe the features or capabilities that the product must have to be viewed as successful.
   - *Concerns:* List any obstacles, constraints, or limiting factors that might impede the project or inhibit stakeholder acceptance of the product.
   - *Technical proficiency:* Describe the direct user's degree of familiarity with the technology.
   - *Work environment characteristics and constraints:* Describe relevant working conditions that might affect system usage (e.g. a noisy work environment or mobile or outdoor usage)
2. **Include the stakeholder profiles in the user requirements document (if used) and the softwre requirements specification document.**
   - If the profiles contain a lot of information, document a profile for each stakeholder as a separate table or section in the appropriate requirementsdocument.

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

Example: Stakeholder profiles

Stake-holder	Roles	Responsi-bilities	Interests	Success Criteria	Concerns	Technical Proficiency/ Work Environment Constraints
CEO	• Sponsor • Indirect user	• Pay for the software project	• Attracting new contractors • Streamlining business operations	• Satisfies office manager's concerns • Reports on how the company is doing	• Adhering to the state and federal tax laws • Having access to the new system as soon as possible	• N/A

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

Elicitation Techniques

Interviews with Stakeholders

What are they?

Face-to-face meetings in which an interviewer asks questions to obtain information from the respondent. Interviews can be unstructured (with no predefined questions) or structured (with questions prepared in advance)

Why do it?

To collect general information about stakeholder needs, to ask customers and users to state their needs, and to help uncover conflicting software requirements.

How to do it?

1. **Identify the people you would like to interview.**
   - Choose a cross section of people. Include sponsors, customers, and users with subject matter expertise.
   - Match the interviewees with interviewers that they are likely to be open with. Be sure that interviewers will be comfortable interviewing senior managers and customers, and vice versa.
2. **Prepare the interview questions.**
   - Clarify the goal of each interview (e.g., to gather background information and high-level features of the software, or to gain a detailed understanding of user work folow or data needs)
   - Construct the interview questions. Sequence them from general to more detailed. Arrange easier, factual questions at the beginning and more difficult interpretive questions later in the interview.
3. **Schedule the interview and arrange the logistics for your meeting.**
4. **Conduct the interview.**
   - Practice active listening and avoid leadingquestions.
   - Be flexible, asking new or follow-up questions as needed
5. **Document the results.**
   - Review your notes immediately after the interview
   - Follow up with the interviewee to resolve any conflicting information.
   - Analyze your notes from multiple interviews to uncover patterns and conflicts

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

What is the difference between Context free questions and Metaquestions?

Ask *contex-free* and *meta-questions*

Context free questions:

- What problems does this system solve?
- What problems could this system create?
- What environment is this system likely to encounter?
- What degree of precision is required or desired in this product?

Metaquestions:

- Am I asking you too many questions?
- Do my questions seem relevant?
- Are you the right person to answer these questions?
- Who else might be able to answer these questions?
- Is there anything else I should be asking you?
- Is there anything you'd like to ask me?

Gause & Weinberg, 1989

Facilitated Workshops

a.k.a. Joint Application Design (JAD), Joint Requirements Planing, Design Workshop, Requirements Workshop,…

What are they? Gathering of carefully selected stakeholders who work together under the guidance of a skilled, neutral facilitator to produce and document requirements models

Why do it? To quickly and efficiently define, refine, prioritize, and reach closure on user requirements. A workshop commits users to the requirements discovery process and promotes user ownership of the deliverables

How to do it?

1. **Determine the workshop's purpose and participants.**
   - Write a concise workshop purpose statement and draft a subset of requirements statements or analysis models before the workshop to use as a starting point.
   - Define the roles (e.g., participants, facilitator, recorder, sponsor, and observers).
   - Keep the workshop small. Strive for a dozen or fewer participants.
   - Use a skilled, neutral facilitator, escpecially if you have a large group or there are many political issues or conflicts involved. Have the facilitator interview some or all of the participants before the workshop, to learn enough to plan the workshop and confirm its purpose.
2. **Identify the workshop's ground rules.**
   - Have the facilitator gather ideas for ground rules from participants at the start of the workshop. Examples: Start and end on time, Be prepared, Focus on interests and not positions
   - Confirm the ground rules and make sure that the participants own and enforce the ground rules.
   - Define decision rules and decision-making process for the workshop
3. **Define the workshop deliverables and design the agenda**
4. **Conduct the session**
5. **Follow up on issues, next steps, and actions**

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

Exploratory Prototypes

a.k.a. Mock-Up, Storyboard

What are they? Partial or preliminary versions of the software created to explore or validate requirements

Why do it? To allow users to give feedback early in the project and actively co-develop requirements with analysts.

How to do it?

1. **Select a portion of the product's scope to prototype.**
   - Choose requirements that are unclear, conflicting, or involve complex user interactions.
   - Choose a small set of functionality.
2. **Determine whether you will create a throwaway prototype or evolutionary prototype.**
   - Clarify the purpose of the prototype with users and team members
   - Establish the technical environment for developing the prototype, if appropriate
3. **Design and build the prototype.**
   - Use real customer data, if possible. When building user interfaces, consider adding example data that would appear to users who are testing the prototype
4. **Conduct the prototype evaluations with users.**
   - Begin with a statement about the goals of the prototype and next steps
   - Demonstrate or simulate a user interacting with the system. Show mock-ups of the top-level interfaces in sequence
   - Record user issues and suggestions
   - Conduct the prototype review in two hours or less
   - Create a summary of the findings and next steps, and agree upon a schedule for the next review
5. **Document the results.**
   - Correct any related models and requirements documents

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

Focus Groups

What are they?

Carefully planned group interviews that raise issues and ask open-ended questions to obtain information from participants. They often consist of a series of meetings between a moderator and groups of six to twelve people, usually with common demographics.

Why do it?

To obtain user reaction to new products or product ideas in a controlled environment, and to reveal subjective information and perceptions about product features. Focus groups explore requirements choices and obtain reactions to new components and interfaces. They also help product development organizations prioritize requirements and identify areas for further qualitative or quantitative study.

How to do it?

1. **Define the objectives and target participants of the focus group.**
   - Decide whether you are looking for a general reaction to new or existing features or if you want to focus on specific capabilities.
   - Determine the target participants and geographic location for the session.
   - Decide how many sessions you will hold and their duration. Each session typically lasts from 90 to 150 minutes, but you should allow time for follow-up questions and undstructured discussions.
   - Develop the questionsto be asked.
2. **Plan and arrange the logistics for the session.**
3. **Conduct the focus group.**
   - Introduce the participants and facilitator, and review the ground rules
   - Obtain consent before you videotape or audio tape the focus group.
   - Summarize the answers as participants address each question
4. **Analyze and document the collected information.**
   - Summarize the results of all of the focus group questions and share the results with the participants
   - Use the information to confirm directions, explore new capabilities or features, determine whether to focus further requirements gathering, or prioritize requirements.

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

Observation

a.k.a. Contextual Inquiry, Field Observation, Ethnography, Shadowing, Soft Systems Analysis, Social Analysis

What are they? Visits by requirements analysts to users' workplaces to watch users perform their jobs. Analysts can as questions to clarify what task the users are performing or why they are performing the tasks; users explain their tasks as they perform their work.

Why do it? To allow analysts and developers to understand how users will use the software in its work context. Observation can surface environmental issues in the users' workplace that will affect requirements. It also uncovers details that might not be obvious when user explain their tasks, because users' work is often intuitive to them and therefore difficult to articulate to others.

How to do it?

1. **Identify the users that you want to observe.**
   - Decide how many users to observe and select users to observe
2. **Arrange for the observation.**
3. **Conduct the observation.**
   - Limit each observation to three hours or less
   - Ask the user interacting with the software to describe what he or she is doing and why
   - Take notes during the observation
   - Ask the user if you can return or call to resolve any follow-up questions that you may have
4. **Analyze and document your observations.**
   - Create and refine user requirements models shortly after each observation, while the information is still fresh in your mind

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

User Task Analysis

a.k.a. Role Modeling, Scenario-Based Engineering, Scripting, Stimulus-Response Sequences, Usage Analysis

What is it?

Using real or made-up examples to describe user tasks and the context within which the work will be performed. Users describe simulated uses of the system by recalling or imagining stories, episodes, or scenarios that they have experienced. Each user task is a stereotypical description, written in text form, of the use of the system to compete a task.

Why do it?

To use examples rather than abstractions to elicit requirements, and to reveal requirements that users may have difficulty recalling outside of their work environment. User task analysis documentation can provide a basis for developing use cases, conducting *model validation*, developing use cases, developing user acceptance tests, and designing prototypes.

How to do it?

1. **Identify and document the user roles.**
   - Survey the user community to understand the types of users, their background, typical work habits and preferences
   - Decide whther you want to do task analysis of the ”as-is” system, the ”to-be” system´, or both
2. **At the meeting, ask users to describe typical tasks that they must accomplish with the system.**
   - Ask for two to five examples of the same or similar tasks, listening for differences in the steps, if any
   - Always address the normal, most typical tasks first, in which no errors or variations occur. After documenting those tasks, explore the alternative steps needed to handle errors or variations
   - Take notes. Don’t try to capture every detail. Ask clarifying questions and repeat the steps or sentences as users describe them.
3. **Document the user tasks.**
   - Write one task step per sticky note or index card and ask users to arrange them in a sequence on a wall, or capture the task information on a laptop and display what is being recorded using a data projector
   - Document a few sentences as a numbered list of four to seven steps for each task. Supplement the task narrative or numbered steps with a visual diagram (such as a flowchart) to show the user’s steps
   - Identify the details that might go into user requirements models
   - Test the task sequences with scenarios to uncover missing steps
   - Review the user profiles from the stakeholder profiles. Check that descriptions of direct users accurately portray typical users now that user tasks are more clearly defined
   - Follow up with users to clarify any unanswered questions
   - Develop related user requirements models and specify the quality attributes

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

Existing Documentation Study

a.k.a. Market Analysis, Requirements Reuse

What is it?

Inspection of existing document sources to uncover requirements information.

Why do it?

To discover or verify requirements using a low-cost technique. A documentation study enables the team to define features provided in a competitor's software and to surface requirements to allocate to people rather than to software. Studying documentation can also provide requirements information when you are replacing an existing system.

How to do it?

1. **Identify the appropriate documentation sources to use.**
   - Ask systems and support staff what documentation exists and whether it is accurate
   - Locate user documentation that could be in physical form (e.g. training manuals and procedural guidelines) as well as in soft form (e.g. help screens and error messages)
   - Search for information on competing products, especially those with functionality that is appealing to customers, most utilized, least utilized, troublesome, or missing
2. **Review and analyze the documentation**
   - Look for patterns that suggest valuable functionality for the new system
   - Search for information about nonfunctional requirements (e.g., performance, usability, and security)
   - Consider which potential requirements you will allocate to people as part of a business process
   - Share and review the findings with customers and users
   - Use the information to identify areas for further exploration and to uncover missing requirements from a set of already draftedrequirements
3. **Create draft analysis models.**

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

Surveys

a.k.a. Market Surveys, Questionnaires

What are they?

A method of gathering information anonymously from a large number of users. A survey can be open-format (permitting respondents to add information on their own and possibly provide unexpected and insightful feedback) or closed-format (with fixed responses, making them faster to answer and easier to analyze).

Why do it?

To inexpensively sample users for their reaction to an existing product or proposed requirements. Surveys allow you to quickly analyze user responses and unobtrusively obtain requirements from users who are generally inaccessible. Surveys can help you obtain subjective information (such as customer's satisfaction with a product or its performance) and information about the relative importance of variousfeatures.

How to do it?

1. **Identify the purpose of the survey.**
   - Determine a discrete goal (e.g. obtaining feedback on proposed features)
2. **Determine the sample group and the method of collection.**
   - For small groups (<150), consider surveying everyone. For a very large group, sample a subset of the user community
   - Decide if your survey will be mailed survey, an online survey, an e-mail survey, a telephone survey, or an on-site survey
3. **Design the survey questions.**
   - Decide if you will use subjective survey questions, objective survey questions, or both
   - Construct unbiased questions. Consider adding slightly different versions of the same questions to verify the repeatability of the responses
   - Ask short, unambiguous questios
   - Be sure that each question addresses a single issue
   - Start with easy questions and group similar questions
4. **Test the survey before you distribute it.**
5. **Administer the survey**
6. **Analyze and document the data**

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

Which techniques to choose?

Techique	Factors to Consider		
	Number of End Users	Accessibility to Subject Matter Experts	Time for Gathering Requirements
Interviews	Not feasible with large numbers of users and experts; use representatives	Requires access to interviewees; can use telephone interviews, although this limits the quality of the information gathered. Interviewers can travel to users who are not at the same location	Total time to conduct interviews, collate findings, and clarify conflicting data can take days or weeks.
Exploratory prototypes	Select one or more representative user(s) from each user group.	Requires direct access to users for prototype reviews unless online tools are used for reviews. Ideally, prototyping is combided with facilitated workshops or user task analysis, which require direct access to users.	Exploratory prototypes that are discarded can be developed in hours, while evolutionary prototypes can take days or weeks. Reviews take only hours, once scheduled. Multiple reviews should be conducted as the prototype is iteratively developed.
Facilitated workshops		Relies on face-to-face interaction to be most effective; usually requires multiple workshops within a short time frame. Users may have to travel to workshop.	Getting the right people for well planned workhsops reduces the time to develop requirements to days or weeks and increases the quality of the requirements.
Focus groups		Relies on face-to-face interaction; usually requires multiple focus group meetings.	Usually takes weeks to plan, conduct, and analyze the data.
Observation		Relies on real-time access to users in their work environment. Observers can travel to user sites.	Can be done over days or weeks, depending on user accessibility.
User task analysis		Relies on face-to-face interaction to be most effective. Usually requires meetings within a short time frame	User meetings followed by documenting the tasks generally take days
Existing documentation study	Not applicable	Not applicable.	Analysis and documentation can take days or weeks
Surveys	Useful for sampling a large number of stakeholders	Physical access not required.	Designing the survey, obtaining responses, and summarizing the data can take weeks or months.

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

Skills and characteristics needed

Technique	Facilitation Skills	Interpersonal Skills	Interviewing Skills	Observing/ Listening Skills	Technical Writing Skills
Interviews		X	X	X	
Exploratory prototypes	X				
Facilitated workshops	X	X		X	
Focus groups	X	X	X	X	
User task analysis		X	X	X	X
Observation		X		X	
Existing documentation study					X
Surveys			X		X

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

Stakeholder Elicitation Plan

a.k.a. Stakeholder Involvement Plan, Stakeholder Inclusion Strategy

What is it? A plan that considers the importance of the various stakeholders' needs and their contributions to the requirements development process.

Why do it? To decide who should be involved in the various requirements activities and how they should contribute. Developing such a strategy helps you avoid overlooking stakeholders and missing requirements. It also help gain commitment from the stakeholders' for their time and involvement.

How to do it?

1. **Rank the importance of each stakeholder in the stakeholder categories. Use ranking scheme such as MoSCoW:**
   - *Must (M)*: Essential to success
   - *Should (S)*: Very important to gather and understand this stakeholder'srequirments
   - *Could (C)*: Good to have this stakeholder'sinvolvement, but less important
   - *Won't (W)*: Not to be considered
2. **Determine how you will involve each stakeholder ranked as an M, S, or C. Consider:**
   - *Degree of involvement:* Decide the extent to which each stakeholder will participate. He or she may fully participate, have some degree of limited involvement, or be indirectly innvolved if a surrogate is representing his or her needs.
   - *Method of involvement:* Determine how the stakeholder will beinvolved:
     - *Actively*: Participates in a requirements workshops, surveys, interviews, focus groups, or prototypes
     - *Passively*: Gets reports from surrogates or reviews e-mail messages about the progress of requirements development
     - *Indirectly*: Supplies help desk or customer request logs or provides anonymous survey or marketing data
   - *Frequencey of involvement*: Decide if the stakeholder will be continuously or periodically involved.
3. **Record the elicitation plan in a table or other document**

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

Example: Stakeholder Elicitation Plan

Stakeholder role	Importance	Degree of involvement	Method of inolvement	Frequency of involvement
CEO	Must (M)	Limited	Passive: Receive status reports via e-mail or short telephone conversations	Periodically: Weekly
Bookkeeper	Must (M)	Full	Active • Interview for wish list requirements • Include in four half-day facilitated workshops to create analysis models • E-mail draft requirements documents	Continuously: Daily during weeks 2-4

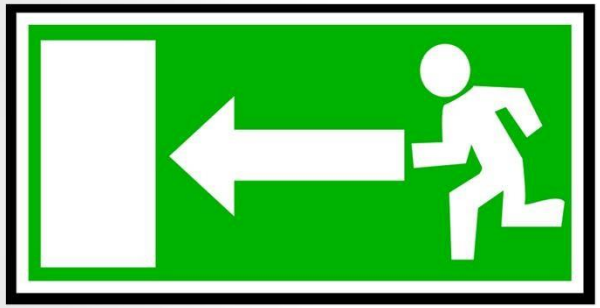
Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

Finding missing requirements

- Decompose high-level requirements into enough detail to reveal exactly what is being requested. A vague, high-level requirement that leaves much to the reader's interpretation will lead to a gap between what the requester has in mind and what the developer builds.
- Make sure that all user classes have provided input. Make sure that each use case has at least on identified actor.
- Trace system requirements, use cases, event-response lists, and business rules in their detailed functional requirements to make sure that the analyst derived all the necessary functionality
- Check boundary values for missing requirements.
- Represent requirements information in multiple ways.
- Sets of requirements with complex boolean logic often are incomplete. Represent complex logic using decision tables or decision trees to make sure you've covered all the possible situations.

K. E. Wiegers, *Software Requirements*. Microsoft Press, 2003.

When to stop elicitation?



You may be done if the users:

- can't think of any more use cases
- propose new use cases but you've already derived the associated functional requirements from other use cases
- repeat issues that they already covered in previous discussions
- suggest new features, user requirements, or functional requirements that are all out of scope
- propose new requirements that are all low priority
- propose capabilities that might be included sometime in the future

K. E. Wiegers, *Software Requirements*. Microsoft Press, 2003.

A university plans to develop a **Sustainability Awareness Application** aimed at educating software engineering students about the environmental impacts of software systems and IT services. The app will feature interactive modules, quizzes, and personalized recommendations for eco-friendly software development practices.

Your team is tasked with gathering detailed requirements to ensure the app meets the following goals:

- Engages students with interactive learning.
- Provides accurate sustainability content.
- Offers tailored recommendations based on user behavior.
- Integrates seamlessly into students' existing digital platforms (e.g., LMS or mobile).

LECTURE 2

Requirements Elicitation

Requirements Analysis

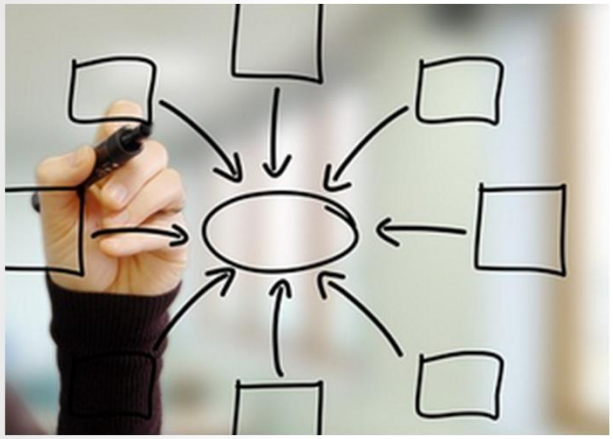
Requirements Prioritization

Requirements analysis: What & Why?

- To effectively analyze requirements, you need to sufficiently understand and define the requirements so that stakeholders can prioritize their needs and allocate requirements to software
- Analysis results in requirements models
  - user requirements represented by diagrams, structured text (lists, tables, or matrices) or a combination
- Analysis also entails prioritizing requirements by analyzing trade-offs among the requirements to make decisions about their relative importance and timelines.
- Requirements analysis is primarily conducted by the Requirements Analysts in close collaboration with key stakeholders

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

Why to model requirements?

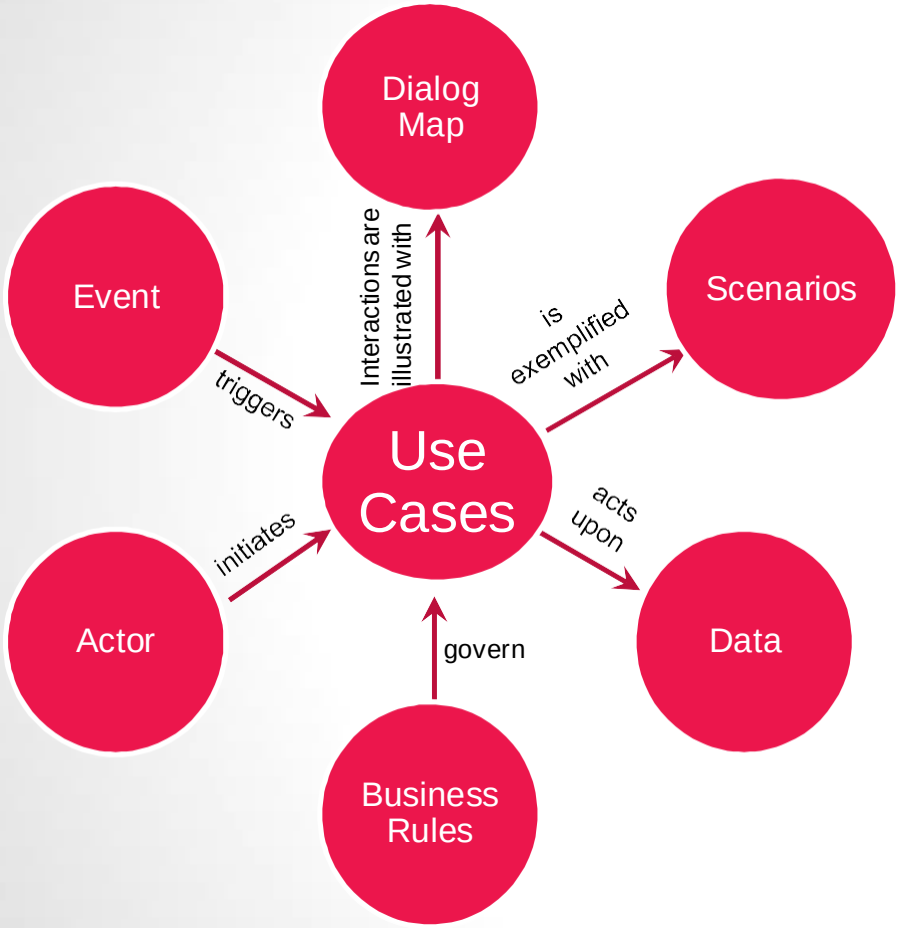


Requirements models will help you:

- Facilitate communication between technical and business people. Models let the team look at different aspects of the user requirements from different perspectives
- Uncover missing, erroneous, vague, and conflicting requirements. User requirements models link together, allowing your team to reveal related requirements and inconsistencies between models. Discovering and correcting these errors results in higher quality requirements
- Make the requirements development process more interesting and engaging to stakeholders. Using both textual and visual models provides variety and permits stakeholders to understand requirements from more than one angle
- Tap into different modes of human thinking. Some people think more precisely with words while others are better able to understand concepts with diagrams. Using both types of representations leverages different thinking modes.

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

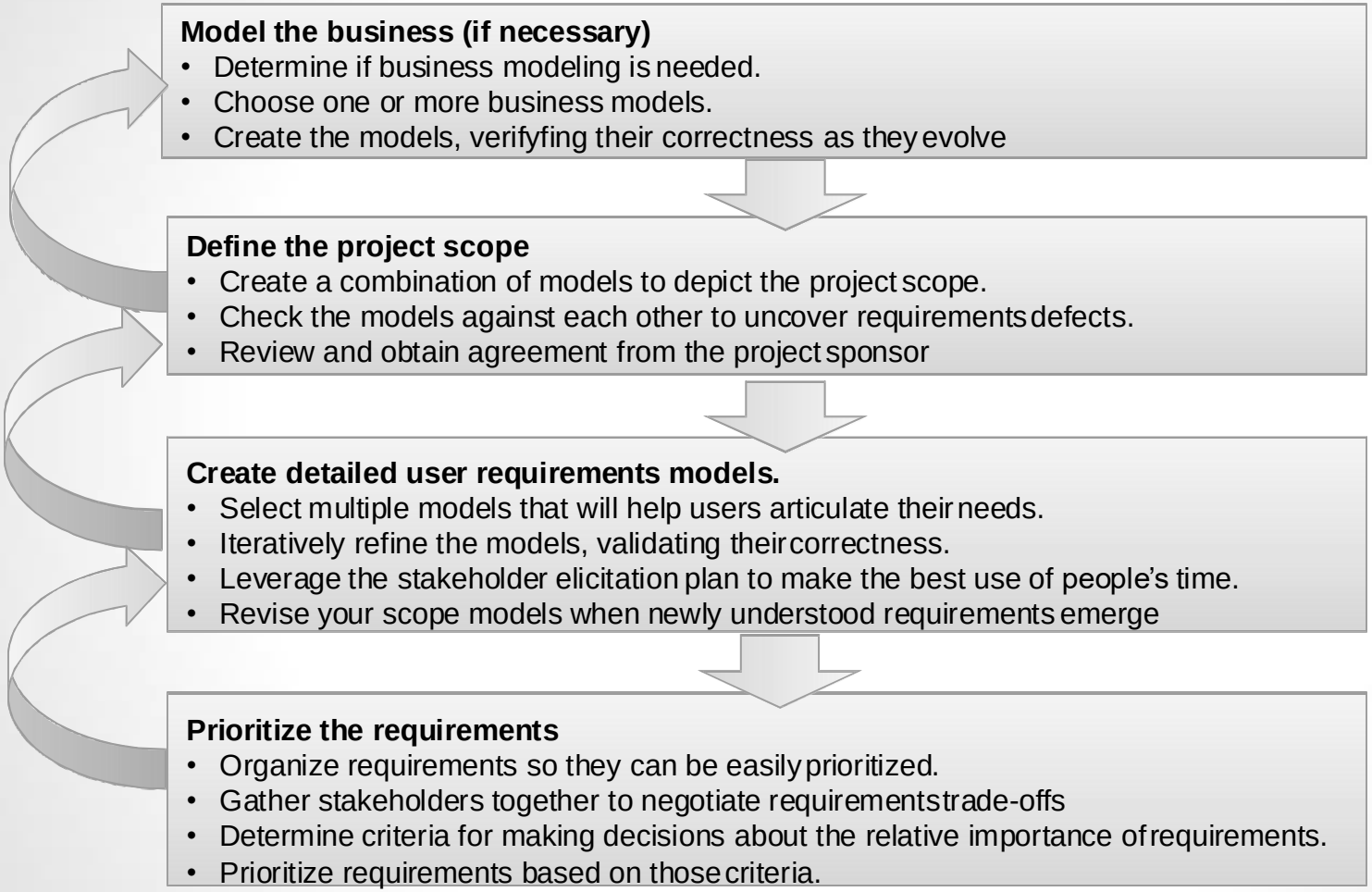
Good User Requirements Modelling Practices



- Each requirements model describes one aspect of a problem
  - No single model can describe all requirements
- Elements of one model link to elements of another
  - Each model can be used to uncover related or missing elements in another model

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

Requirements Analysis Cycle



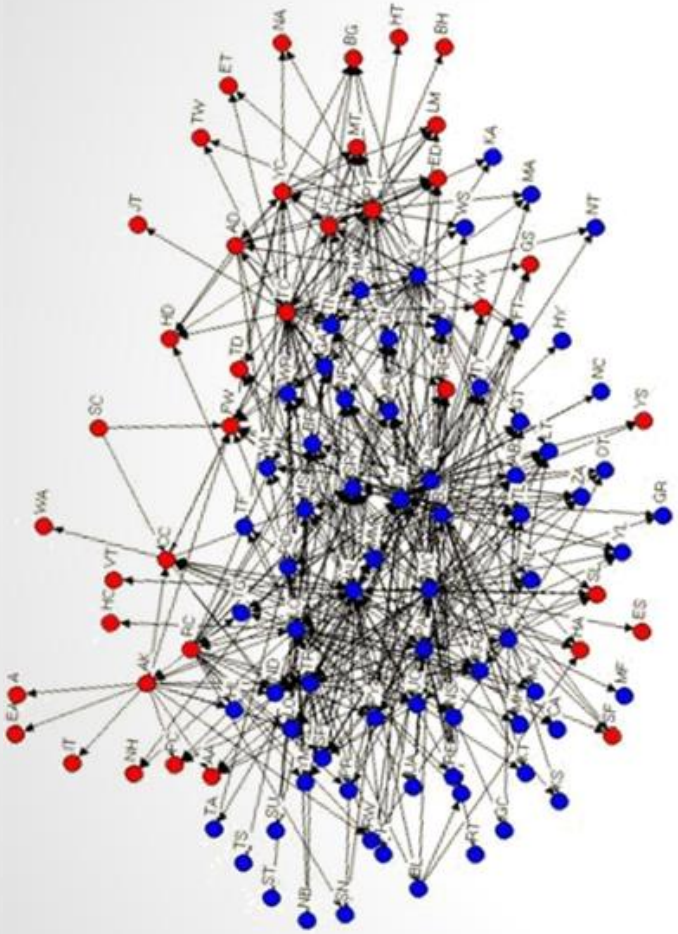
Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

From Customer's Voice to Analysis Model Components

Type of Word	Examples	Analysis Model Components
Noun	• People, organizations, software systems, data items, or objectys that exit	• Terminator or datastores (DFD) • Actors (use-case-diagram) • Entities or their attributes (ERD) • Classes or their attributes (class diagram)
Verb	• Actions, things a user can do, or events that can take place	• Processes (DFD) • Use Cases (use-case diagram) • Relationships (ERD) • Transitions (STD) • Activities (activity diagram)

A **chemist** or a member of the **chemical stockroom staff** can *place* a **request** for on ore more **chemicals**. The request can be *fulfilled* either by *delivering* a **container** of the chemical that is already in the **chemical stockroom's inventory** or by *placing* an **order** for a new container of the cheemical with an outside **vendor**. The **person** placing the request must be able to *search* **vendor catalogs** on line for specific chemicals while *preparing* his or her request. The system needs to *track* the **status** of every chemical request from the time it is prepared until the request is either fulfilled or *canceled*. It also eeds to track the **history** of every chemical conainer from the time its *received* at the **company** until it is fully *consumed* or *disposed* of"

Create Readable Diagrams



Diagrams can become complex and difficult to read.

Some advices to avoid this:

- Allow for additional space around the diagram so that information is not crowded together or difficult to read
- Break larger models into multiple pages to increase readability
- Organize the diagrams for readability, from left to right, and from top to bottom.
- Minimize lines crossing over symbols or other lines, which can be difficult to read and confusing to understand
- Show only what is important, keeping the diagram simple. Selectively show details for areas that are particularly complex or controversial
- Do not use all modeling elements just because they exist
- Use naming conventions and glossary terms consistently across the diagrams
- Focus on the accuracy and correctness of the diagram, not its beauty and comprehensiveness.

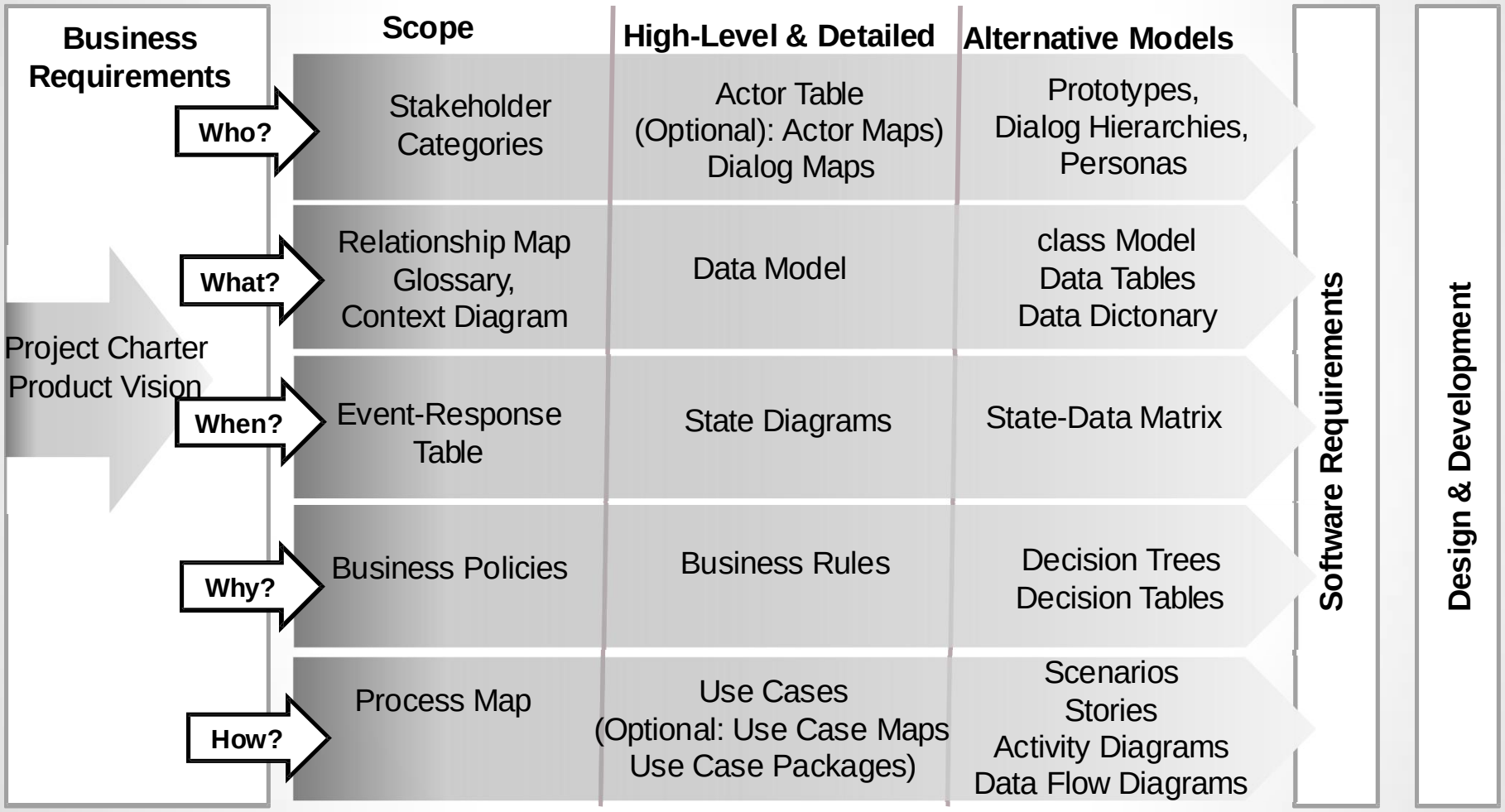
Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

What models to create?

Focus Question	Example Questions	User Requirement Models for this Focus
Who	• Who are the project's stakeholders? • Who will directly interact with the system? • Who will see what when they interact with the system?	• Stakeholder categories • Actor table (and possibly an actor map) or personas • Dialog map (supplemented with or substituted by a prototype or dialog hierarchy)
What	• What do important busines term mean? • What functions in the organization interact to share information? • What information or assets go into and out of the system? • What are the static data elements that must be stored and how are they related?	• Glossary • Relationship map (a business model) • Context diagram • Data model (supplemented with or substituted by a class model, data tables, or a data dictionary)
When	• When does the system need to respond or act? • When do tasks get performed and when does information change?	• Event-response table • State diagram (supplemented with or substututed by a state-data matrix)
Why	• Why are we motivated to enforce standards, policies, regulations, and legislation? • Why are the decisions made that influence behavior and assert business structure?	• Business Policies • Business rules (supplemented with or substituted by decision tables or decision trees)
How	• How do processes operate in the business to achieve business goals? • How are tasks performed and in what sequence?	• Process map (a business model) • Use cases and possibly use case maps and use case packages (supplemented with or substituted by scenarios, stories, activity diagrams of use cases, or data flow diagrams)

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

User Requirements Models Roadmap



Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

Selecting Tools & Techniques

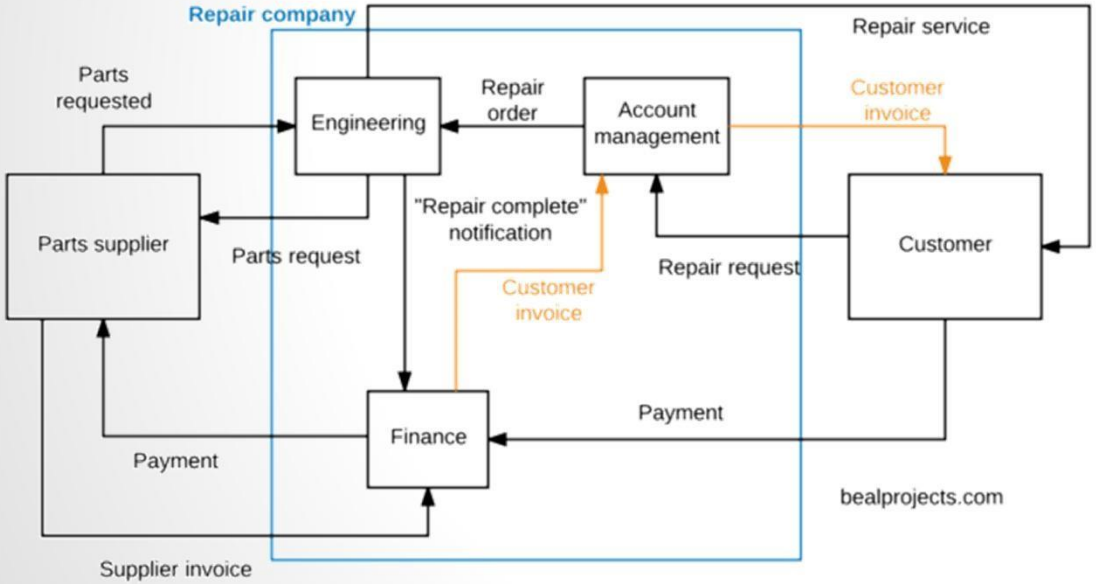
When you need to:	Then create:
Model the Business	Some combination of Relationship Map and/or Process Map
Understand the project scope	Some combination of Context Diagram, Event-Response Table, and/or Business Policies
Add detail to user requirements	Some combination or variation of Actor Table, Use Cases, Dialog Maps, Data Model, State Diagrams, and/or Business Rules
Negotiating trade-offs among requirements	Prioritized Requirements

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

Relationship Map

a.k.a. Business Interaction Model, Organizational Context Diagram, Organizational Relationship Map

Shows what information and products are exchanged among external customers, providers, and key functions in the organization



https://bealprojects.com/products/when-visual-models-are-better-than-textual-requirements-part-ii-relationship-maps/

How to do it?

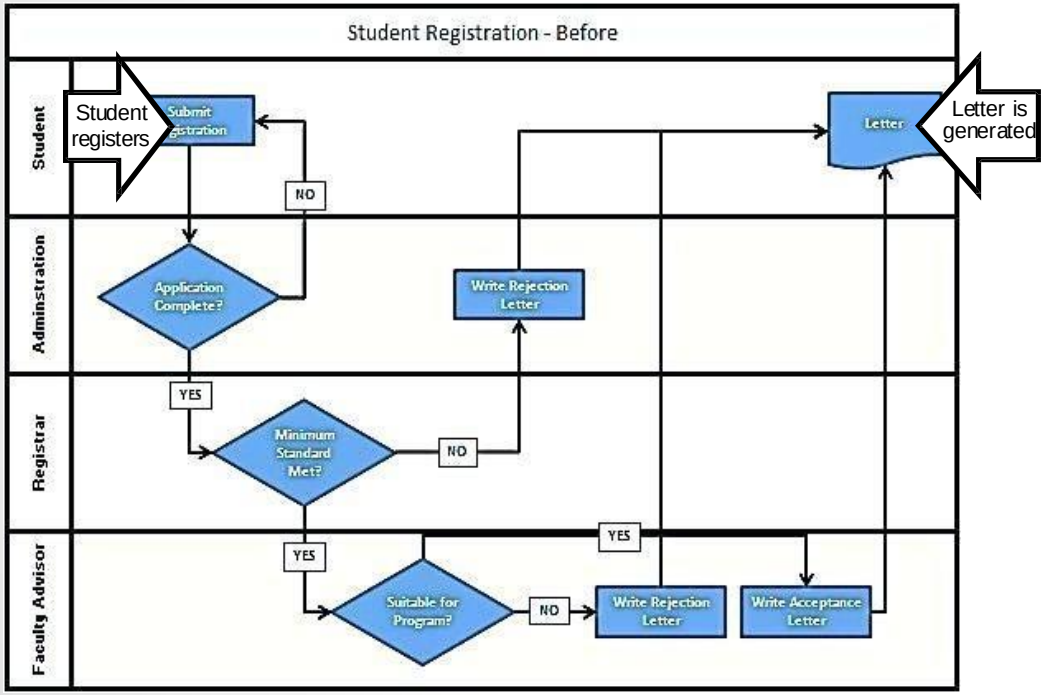
1. Draw the key functions, departments, and work groups involved in the business process as boxes
   - Choose business functions logically, not necessarily according to an organizational chart
   - More detail is typically better
2. List the key inputs and outputs that each function receives or produces
3. Connect the inputs and outputs to the functionst that use and produce them
   - Use arrowheads to indicate the flow direction
   - Label each arrow with the name of the input or output

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

Process Map

a.k.a. Swimplane Diagram, Cross-Functional Process Map, Line of Visibility Model

Shows the sequence of steps, inputs, and outputs needed to handle a business process across multiple functions, organizations, or job roles



http://dcdesigns.info/swim-lane-diagram-template/swim-lane-diagram-template-swim-lane-diagram-template-excel-example-before-classy-swim-lane-diagram-template-ppt/

How to do it?

1. Name the business process to be modeled, starting with an action verb
2. Define the *business event* that triggers or starts the process
3. Name the end point or outcome of the process
4. List the participants in the business process in a column along the left side of the diagram
5. Create horizontal rows or ”lanes” for each participant, to represent the organizational entity or role where the work is done
6. Identify all of the process steps that occur between the triggering event and the outcome
7. Identify the outputs of each step
8. Review the diagram and revise it as needed.

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

Event-Response Table

a.k.a. Evebt Table,Event List

Identifies each event (i.e., an input stimulus that triggers the system to carry out some function) and the event responses resulting from those functions.

Event-Response Table: Windshield Wipers

Event	System State	System Response
set wiper control to low speed	wiper off or wiper on high speed or wiper on intermittent	set wiper motor to low speed
set wiper control to high speed	wiper off or wiper on low speed or wiper on intermittent	set wiper motor to high speed
set wiper control set to off	wiper on high speed or wiper on low speed or wiper on intermittent	complete current wipe cycle; turn wiper motor off
set wiper control to intermittent	wiper off	read wipe time interval setting; initialize wipe timer
set wiper control to intermittent	wiper on high speed or wiper on low speed	read wipe time interval setting; complete current wipe cycle; initialize wipe timer
wipe time interval has passed since completing last cycle	wiper on intermittent	perform one low-speed wipe cycle
change intermittent wiper interval	wiper on intermittent	read wipe time interval setting; initialize wipe timer
change intermittent wiper interval	wiper off or wiper on high speed or wiper on low speed	no response
immediate wipe signal received	wiper off	perform one low-speed wipe cycle

In Search of Excellent Requirements 73 *Copyright © 2011 Karl E. Wiegers*

https://uit.stanford.edu/sites/default/files/images/2017/08/30/Event%20Response%20Table.png

How to do it?

1. Name the events
2. For each event, describe its required response
3. Verify the event-response table against existing models and revise it as needed

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

Business Policies

a.k.a. Regulations, Legislation, Standards

Guidelines, standards, and regulations that guide or constrain the conduct of a business.

Policy Group	Policy Identifier	Policies	Owner	Sources
Discounting	BP-1	Provide discounts to senior citizens	Jim Bean, Marketing Director	Pricing guidelines in Marketing / Pricing folder
	BP-2	Provide discounts to military personnel.		

How to do it?

1. Identify groupings of business policies for the problem domain
   - Look at regulations, standard operating procedures, training manuals, ….
   - Be sure that policies align with one or more business goals or objectives
   - Group policies into like groups and name each group to include many related policies
2. Determine where you will allocate the policies
   - manually enforced or implemented by the software?
3. Document policies that are in scope for the project
   - Document policies and uniquely label each policy
   - Consider identifying candidate attributes such as owner, origin, source, volatility. Select only those attributes that you are willing to track and that serve some useful purpose.

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

CAPTURING BUSINESS RULES

Type	Description	Example
Facts	Simple statements that are true about the business	*Every chemical container has a unique bar code identifier*
Constraints	Restrict the actions that the system or its users may perform	*A borrower who is less than 18 year old must have a parent or a legal guardian as cosigner on the loan*
Action Enablers	A rule that triggers some activity under specific conditions is an action enabler	*If the chemical stockroom has containers of a requested chemical in stock, then offer exiting containers to the requester*
Inferences	A rule that establishes some new knowledge based on the truth of certain conditions.	*If the payment is not received within 30 calendar days of the date it is due, then the account is delinquent*
Computations	Many computations follow rules that are external to the enterprise, such as income tax withholding formulas	*The unit price is reduced by 10% for orders of 6 to 10 units, by 20% for orders of 11 to 20 units, and by 35% for orders of more than 20 units*

K. E. Wiegers, *Software Requirements*. Microsoft Press, 2003.

Business Rules

Textual statemens that decompose business policies. Business rules describe what defines, constrains, or enables the software behavior.

How to do it?

1. Identify the sources of business rules
2. Ask questions to identify business rules
   - Focuse on what needs to be known, calculated, decided, triggred or constrained
   - Document each business rule in natural language statements
3. Document the rules
   - Determine whether each rule will be enforced in software or implemented by people
   - Identify useful attributes about business rules
   - Identify relationships between business rules and other user requirements
4. Analyze the rules for consistency and necessity

Business Rule Group	Business Rule Identifier	Business Rule	Business Rule Category	Effective Date	Use Cases	Source
Maintaining Customers	BR-2	If a Customer pays for more than one site, he is considered a Commercial Customer.	Interference	July4	UC 2 (Set Up Customer) UC 4 (Provide Phone Estimate)	Jim Bean, Marketing Director
Invoicing	BR-18	If a Repeat Customer's Jobs exeed 5000€ in a continuous 12-month perioed, offer 15% discount	Action enabler	July4	UC 4 (Provide Phone Estimate) UC 13 (Pay for Job)	Pricing Guidelines Version 5.6

e

Actor Table

a.k.a. Actor Catalog, Actor Description, User RoleModel

Identifies and classifies system users in terms of their roles and responsibilities

Actor	Attributes and Responsibilities	Job Title(s)
Scheduler	Find available Contractors for a Customer's request that match the location. Arrange for services by Contractors at Customer location on requested days and times.	Scheduler, Office Manager
Job Closer	Reconcile Estimated and Scheduled Jobs with Completed Jobs. Apply payments. Issue invoices for Completed but Unpaid Jobs. Update Customer details for changes in site conditions	Bookkeeper, Office Manager, Scheduler

How to do it?

1. List the roles played in the system and place the role name in the "Actor" column of the table
   - Draw upon the direct users from the stakeholder classes and the external entities on the context diagram
2. Place attributes for each actor in additional columns
   - Write a brief description of each actor's responsibilities
   - Add additional columns to hold other attributes
3. Review the actor table for missing or extraneous actors.

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

Use Cases

a.k.a. Evebt Table,Event List

Use Case ID:	6.0		
Use Case Name:	Approve Sales Forecast		
Created By:	Stephanie Famuyide	Last Updated By:	Joan Albert
Date Created:	July 20, 2013	Date Last Updated:	August 10, 2013

Actor(s):	Sales Director
Description:	The sales director approves forecasts for every SKU to confirm that the values in the approval list are realistic.
Preconditions:	• User credentials have been validated • User has approval rights • Sales forecasts have been aggregated
Postconditions:	• User saves record of approval • Sales forecasts are committed for production planning • User navigates away
Normal Course:	• User logs in • User approval list is generated • User performs an approval list action
Alternative Courses:	2. If approval list is marked as incomplete, display message and return user to home screen 3. If approval list is approved, send forecasts for production planning. 3. If approval list is rejected, notify demand planner 3. If approval list has an amendment request, notify demand planner to make necessary changes
Exceptions:	None
Includes:	
Priority:	High
Frequency of Use:	Once per month
Business Rules:	Sales Forecast are to be approved before use in production planning
Special Requirements:	
Assumptions:	Nil
Notes and Issues:	
Use Case Graphic:	

Descriptions in abstract terms of how actors use the system to accomplish goals. Each use case is a logical piece of user functionality that can be initiated by an actor and described from the actor's point of view in a technology-neutral manner

How to do it?

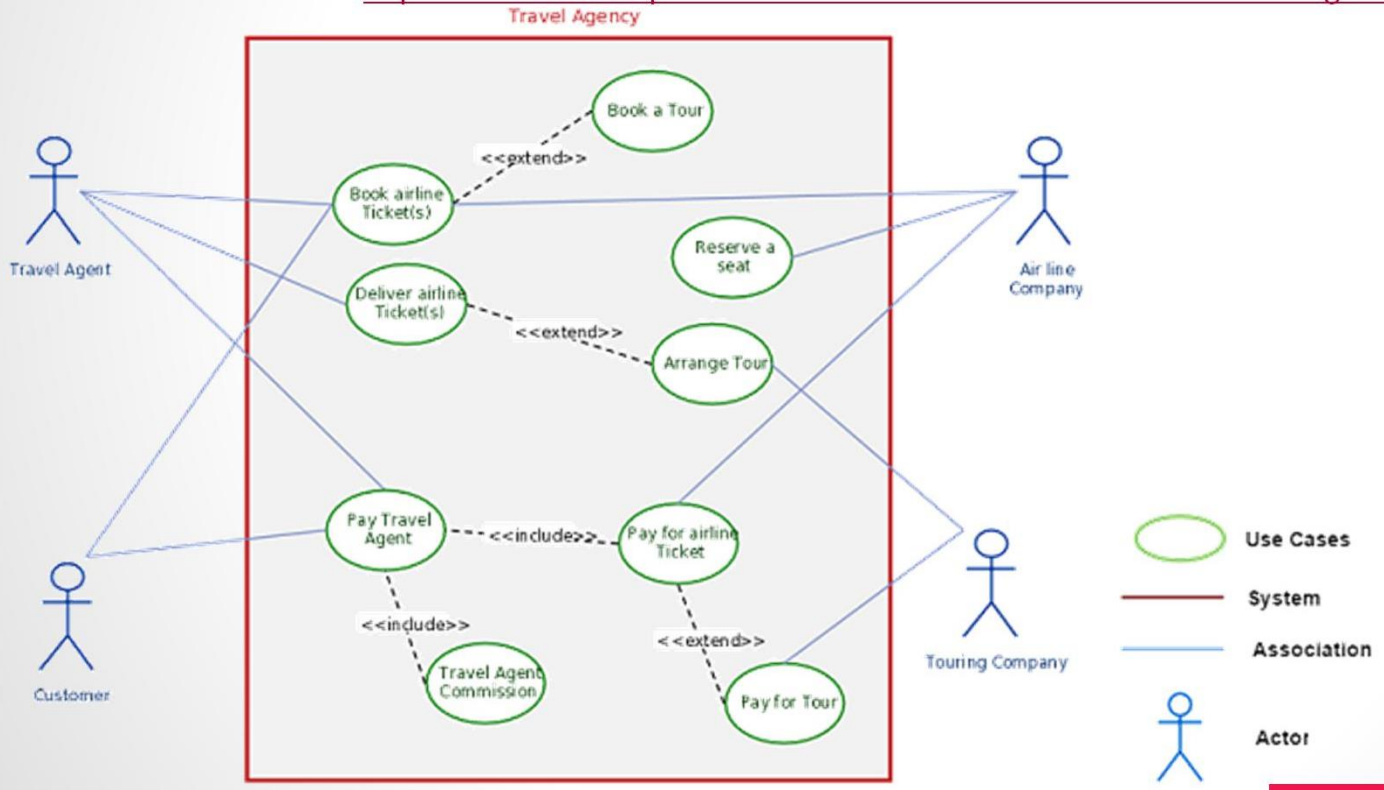
1. Create an initial list of use cases
2. Create a brief description for each use case
3. Create a header for each use case
4. Verify the initial set of use cases before adding details to each
5. Determine the steps in the use cases
6. Test the steps for sequence and completeness
7. Determine the steps for use case exceptions
8. Document steps for any use case variations
9. Identify and separately document *included use cases*
10. Check use cases for missing requirements
11. Define attributes to associate to use cases

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

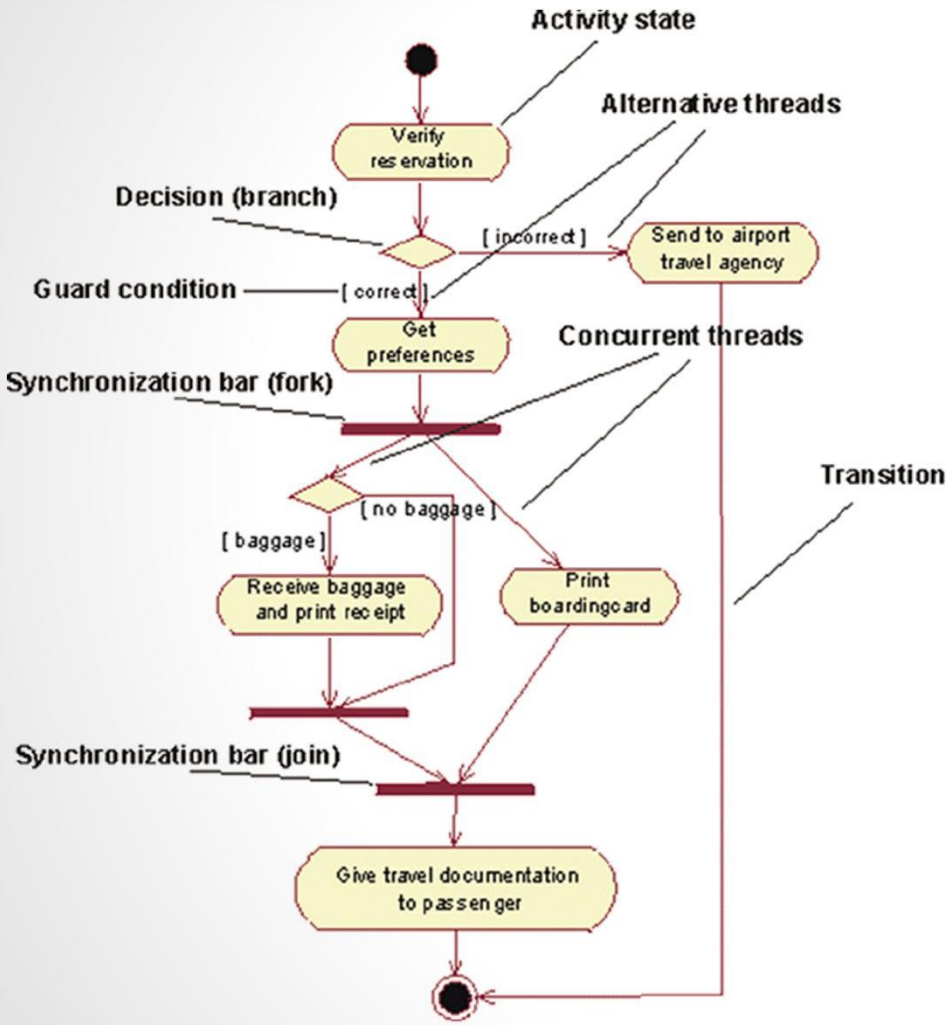
Use Case Diagram

Behavior diagrams which are used to define a set of actions that system should or can perform in collaboration with one or more external users of the system (actors). Each use case should provide some observable and valuable result to the actors or other stakeholders of the system.

Read more from: https://www.tutorialspoint.com/articles/how-to-create-a-use-case-diagram



Activity Diagram

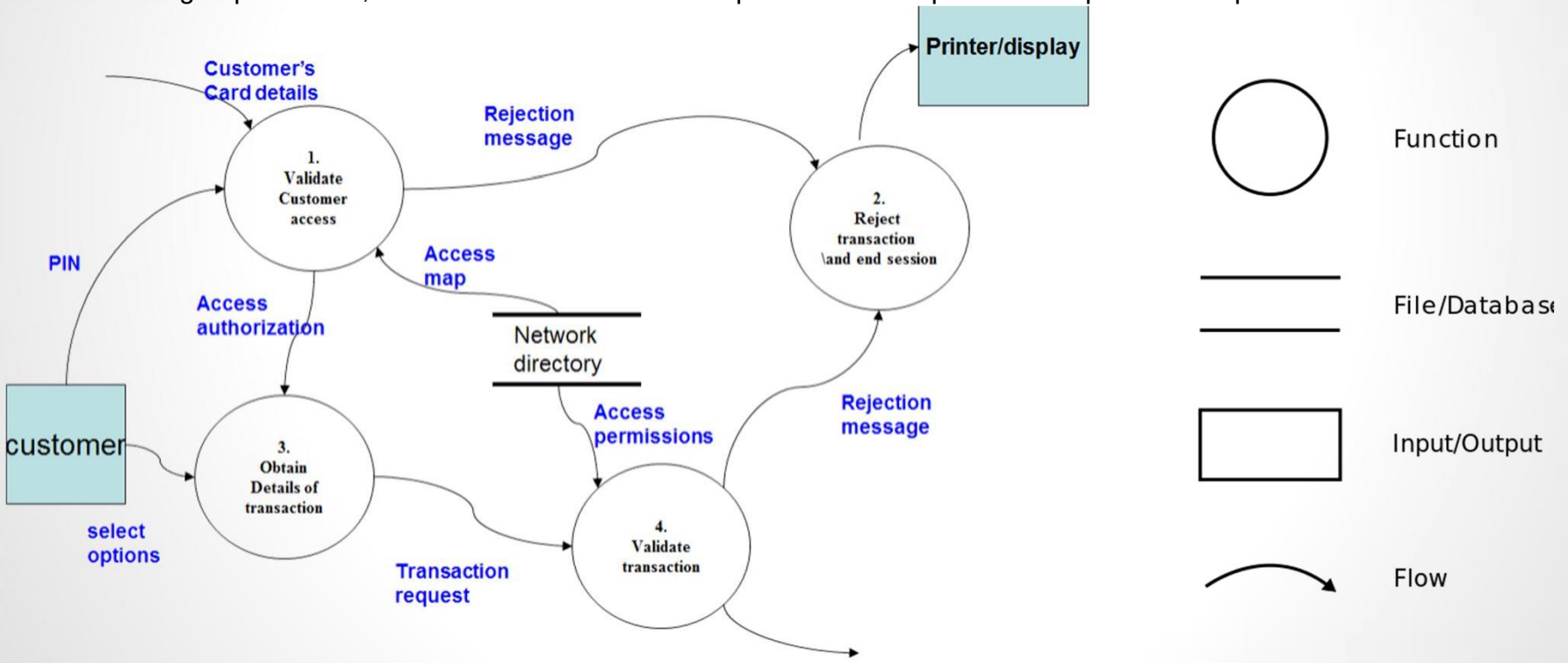


Illustrates the flow of complex use cases using UML notation. It is useful for showing the use case steps that have multiple extension steps, and for visualizing use cases

Read more from:
https://www.ibm.com/developerworks/rational/library/2802.html

Data Flow Diagram

A graphical representation of the *"flow" of data through an information system*, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). A DFD shows what kinds of information will be input to and output from the system, *where the data will come from and go to, and where the data will be stored*. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.

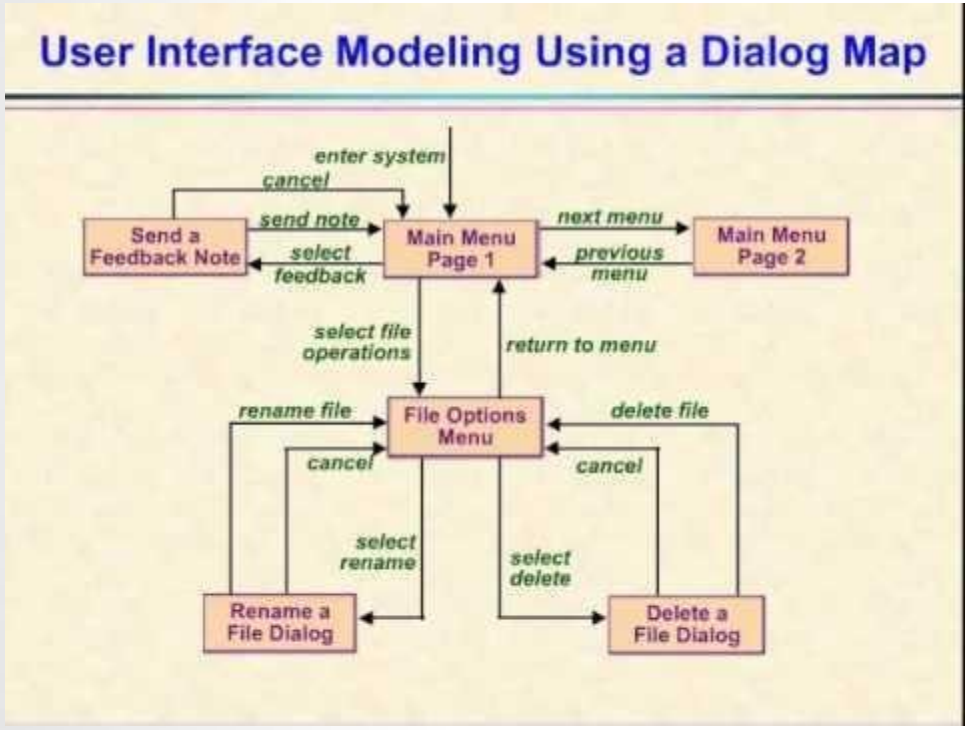


https://stackoverflow.com/questions/23185771/how-to-convert-a-data-flow-diagram-dfd-to-activity-diagram?rq=1

Dialog Map

a.k.a. Context Navigation Map, Storyboard, User Interface Flow Diagram, User Interface Navigation Diagram

Illustrate the architecture of the system's user interface. The dialog map shows visual elements that users manipulate to step through tasks when interacting with the system



How to do it?

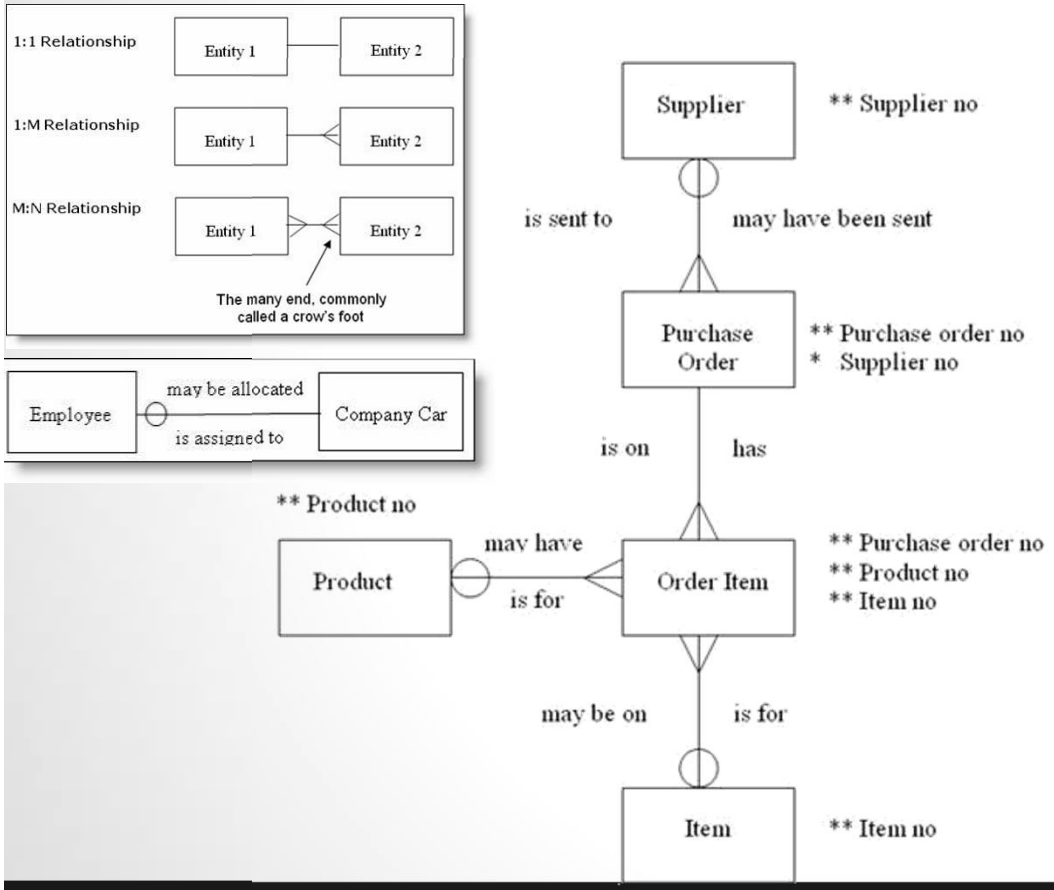
1. Select a single complex use case or a set of related use cases
2. Choose dialogs and identify transitions among them
   - Identify dialogs and represent each dialog as a box
   - Identify transitions from one dialog to another, including user-generated and system-generated triggers.
3. Draw the diagram using the appropriate symbols and transition labels
   - Draw each transition between dialog boxes
   - Add arrowhead pointing to the dialog into which the transition occurs
4. Verify the dialog map for completeness, correctness, and concistency.

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

Data Model

a.k.a. Conceptual Data Model, Entity Relationship Diagram (ERD), Logical DataModel, Domain Model

Shows the informational needs of the system by illustrating the logical structure of data independent of the data design or data storage mechanism.



How to do it?

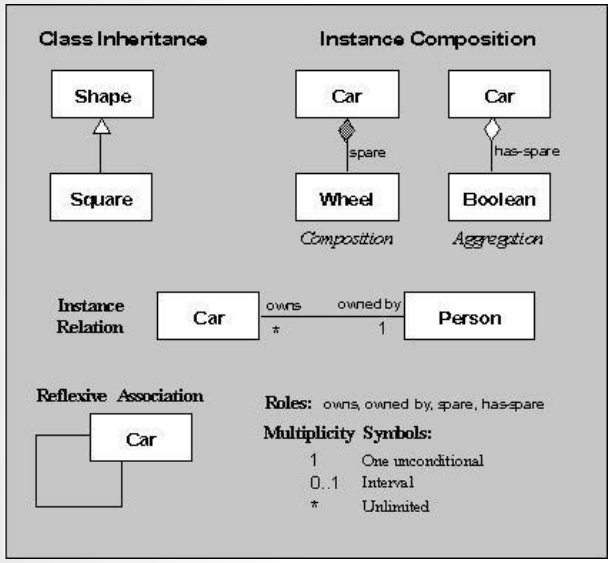
1. Identify and define data entities
2. Define a primary key for each entity
   - Add the primary key to the diagram by writing the key at the top of the attribute list and underlining it.
3. Modify relationships among the entities
4. Identify and diagram the *cardinality* and *optionality* for each relationship
5. Verify the data model

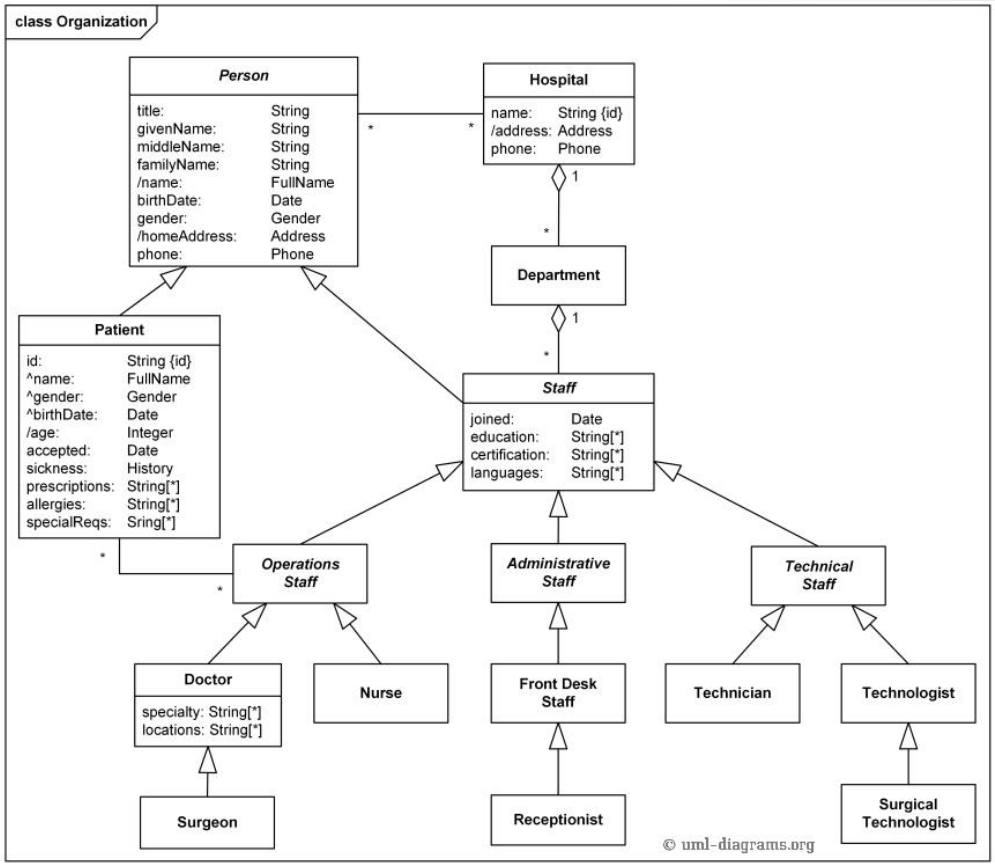
Read more from:
https://www.sqa.org.uk/e-learning/SoftDevRDS02CD/page_42.htm

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

Class Model

Collection of similiar objects (i.e. person, places, events, and physical artifacts). Use class model for projects employing object-oriented software development methods, tools, or databases.

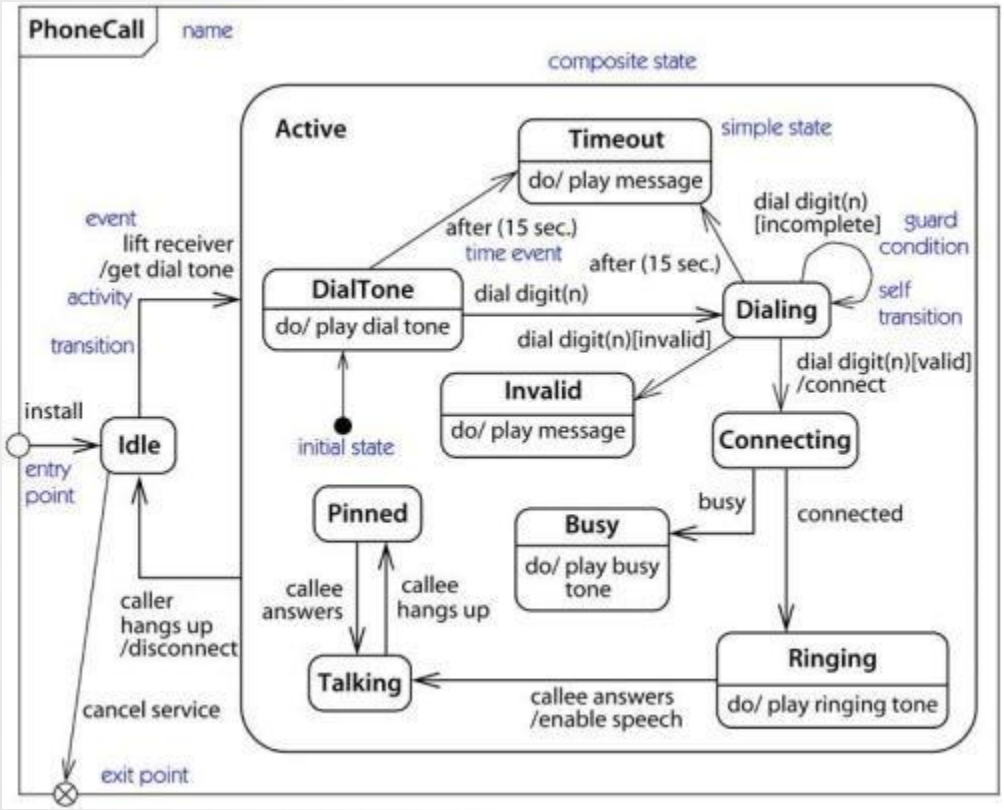




State Diagram

a.k.a. Statechart Diagram, State TransitionDiagram

Visual representation of the lifecycle of a data entity. Events trigger changes in data, resulting in a new state for that entity.



How to do it?

1. Select the critical entities from the datamodel
2. For each selected entity, list possiblelife cycle states
3. Reduce the list of states to those that apply tothe product vision and the functionlity within the product's scope
4. Arrange the states in time-ordered sequence.
5. Identify triggering events for each transition
6. Review related requiremetns for missing elements

Ellen Gottesdiener (2005): The Software Requirements Memory Jogger

LECTURE 2

Requirements Elicitation

Requirements Analysis

Requirements Prioritization

Why Prioritize Requirements?

A way to deal with competing demands for limited resources

- Enables you to provide the highest value at the lowest cost
- Critical for timeboxed or incremental development with tight, immovable release schedules
- Enables project manager to balance the desired project scope against the constraints of schedule, budget, staff, and quality goals

K. E. Wiegers, *Software Requirements*. Microsoft Press, 2003.

Typical Prioritizatin Challenges

- It is hard to quantify the difference between requirements
  - Relative difference is easier to say
- Not all requirements can be compared
- Requirements are on different abstraction levels
- Requirements have dependencies
  - e.g. implementing one requirement is useful only if another requirement is also implemented
- Stakeholders disagree

When customer thinks all requirements are of high priority

Useful questions to ask:

- Is there some other way to satisfy the need that this requirement addresses?
- What would the consequence be of ommitting or deferring this requirement?
- What would the impact be on the project's business objectives if this requirement weren't implemented immediately?
- Why would a user be unhappy if this requirement were deferred to a later release?

K. E. Wiegers, *Software Requirements*. Microsoft Press, 2003.

Some ways of prioritizing (1 of 3)

https://businessanalystlearnings.com/blog/2016/8/18/a-list-of-requirements-prioritization-techniques-you-should-know-about

- **NUMERICAL ASSIGNMENT (GROUPING).** Group requirements into different priority groups with each group representing something stakeholders can relate to.
  - Groups should be clearly defined so that stakeholders do not have a different understanding of each during the prioritization exercise.
  - To prevent stakeholders from putting all requirements in one category, the percentage of requirements that can be placed in each group should be restricted. One disadvantage to this, however, is the fact that requirements in each group will then have the same priority with no unique priority assigned per requirement.
  - For example, requirements can be grouped into *critical*, *moderate* and *optional* priority. Stakeholders may also classify requirements as *compulsory*, *very important*, *rather important*, *not important*, *and does not matter* in order to describe their importance.
- **Five Whys.** It often happens that stakeholders want to implement a certain feature for reasons that are not founded on logical arguments or the business interests of the company. With five whys, the analyst asks the stakeholder repeatedly (five times or less) why the requirement is necessary until the importance of the requirements is established. The answers reveal whether the requirement is really necessary or can be cancelled/postponed once the priority is determined.

Some ways of prioritizing (2 of 3)

https://businessanalystlearnings.com/blog/2016/8/18/a-list-of-requirements-prioritization-techniques-you-should-know-about

- **Analytic Hierarchy Process (AHP).** Stakeholders decompose their goal into smaller sub-problems, which can easily be comprehended and analyzed (in the form of a hierarchy). Once the hierarchy is built, decision-makers evaluate the elements by comparing pairs to each other. The total number of comparisons recommended with AHP are n × (n-1)/2 (where n is the number of requirements) at each hierarchy level. Participants make judgements (sometimes based on data) about the relative importance of each element. Numerical values (based on priorities) can then be assigned to each element of the hierarchy. This method is not suitable for a high number of requirements as the number of requirements determine the number of comparisons that need to be made.
- **MoScoW Technique.** Instead of numbers, stakeholders can prioritise requirements in a collaborative fashion using four priority groups:
  - MUST (Mandatory), SHOULD (Of high priority), COULD (Preferred but not necessary), WOULD (Can be postponed and suggested for future execution)
- **Bubble Sort Technique.** Take two requirements and compare them with each other. If you find out that one requirement should have greater priority over the other, you swap them accordingly. You then continue in this fashion until the very last requirement is properly sorted. The result is a list of requirements that are ranked.

Some ways of prioritizing (3 of 3)

https://businessanalystlearnings.com/blog/2016/8/18/a-list-of-requirements-prioritization-techniques-you-should-know-about

- **Hundred Dollar Method.** This simple method is useful anywhere multiple stakeholders need to democratically vote on which requirements are the most important. All stakeholders get a conceptual 100 dollars, which they can distribute among the requirements. As such, the stakeholder may choose to give all 100 dollars to a single requirement, or the person may distribute the points more evenly. The higher the amount allocated to each requirement, the higher the priority of the requirement. At the end, the total is counted and the requirements are sorted based on the number of points received.

  This technique should only be used when you have a small group of requirements to prioritize and when you have the same set of requirements to prevent respondents from influencing their results by assigning more dollars to their favourite requirement.

- **RANKING.** Give each requirement a different numerical value based on its importance. For example, the number 1 can mean that the requirement is the most important and the number *n* can be assigned to the least important requirement, *n* being the total number of requirements. This method works best when you are dealing with a single stakeholder as it can be difficult to align different stakeholders' perspectives on what the priority of a requirement should be; taking an average can however, address this problem to some extent.

Matrix Prioritization (Example)

http://bawiki.com/wiki/techniques/matrix-prioritization/

Stakeholders	Sales										Marketing										Implementation Team					
Feature Name	Relative Benefit	Benefit Weighting	Relative Penalty	Penalty Weighting	Volatility	Volatility Weighting	Urgency	Urgency Weighting	Criticality	Criticality Weighting	Relative Benefit	Benefit Weighting	Relative Penalty	Penalty Weighting	Volatility	Volatility Weighting	Urgency	Urgency Weighting	Criticality	Criticality Weighting	Implementation Effort	Effort Weighting	Implementation Confidence	Dependency Risk	Risk Weighting	Priority Value
Customizable Interface	3	2	3	1	4	1	2	2	1	1	2	1	1	1	4	1	1	1	2	1	5	2	3	4	1	0.48
Custom Contact Attributes	2	2	3	1	4	1	3	2	2	1	3	1	0	1	4	1	0	1	0	1	2	2	4	3	1	0.67
Multi-Lingual	2	2	1	1	4	1	1	2	1	1	3	1	1	1	4	1	2	1	1	1	2	2	5	4	1	0.47
Email Marketing	1	1	3	1	2	1	2	2	3	1	4	2	4	1	2	1	4	1	4	1	4	2	2	1	1	1.48
Time-Tracking	4	2	3	1	5	1	4	2	3	1	2	1	0	1	5	1	2	1	2	1	1	2	2	3	1	1.65
Pipeline Tracking	3	2	3	1	4	1	4	2	4	1	3	1	2	1	4	1	2	1	2	1	3	2	2	2	1	1.36
Lead Tracking	2	1	3	1	3	1	4	2	4	1	3	2	2	1	3	1	4	1	3	1	3	2	2	3	1	1.52
Email Integration	5	1	4	1	4	1	3	2	3	1	3	1	1	1	4	1	3	1	3	1	2	2	4	3	1	1.04
Calendar Integration	5	2	4	1	5	1	4	2	2	1	2	1	1	1	5	1	3	1	3	1	2	2	4	3	1	1.14
Campaign Management	1	1	2	1	2	1	3	2	4	1	4	2	5	1	2	1	5	1	5	1	4	2	1	0	1	3.00
Desktop Access	5	1	5	1	5	1	3	2	5	1	3	1	4	1	5	1	5	1	5	1	0	2	5	3	1	2.92
Web-Based Access	3	1	1	1	4	1	2	2	0	1	2	1	1	1	4	1	0	1	0	1	3	2	4	3	1	0.31
Mobile Access	3	1	2	1	2	1	3	2	3	1	2	1	1	1	2	1	0	1	0	1	6	2	3	3	1	0.40
Overall Ease of Use	5	2	4	1	2	1	4	2	3	1	4	2	4	1	2	1	4	1	3	1	3	2	3	3	1	1.76
Project Management	0	1	0	1	2	1	0	2	0	1	2	1	2	1	2	1	2	1	0	1	2	2	2	2	1	0.43
Task Management	4	1	5	1	5	1	4	2	3	1	2	1	3	1	5	1	3	1	2	1	1	2	2	2	1	1.88
Workflow Automation	4	2	4	1	3	1	3	2	4	1	2	1	4	1	3	1	3	1	3	1	4	2	1	1	1	2.27
Analytics Engine	1	1	2	1	1	1	3	2	5	1	4	2	5	1	1	1	5	1	5	1	5	2	2	1	1	1.61