PREVIOUS < Free Open Study > NEXT

Requirements Elicitation Methods

We will look at interviewing, brainstorming, mind mapping, facilitated application specification technique (FAST), Joint Application Design (JAD), and use case scenarios as viable methods for eliciting software requirements. These are some suggested activities and thoughts that are common to all:

- Actual users need to be involved in the requirements-gathering process rather than surrogates.
- All stakeholders should be identified; a representative from each type of stakeholder should be involved in requirements gathering.
- Ambiguous requirements should be identified as candidates for prototyping.
- Customers and sponsors are not always the users of the delivered application, but they are the entities that pay for the requirements-gathering work, as well as for the final system.
- Domain constraints that could limit the functionality or performance of the system or software under development should be identified.
- Each stakeholder will have a natural bias, including one toward his or her organization.
- Focus groups are not part of the methodology, as they are not composed of decision-making individuals.
- If only software is under development, the technical environment (e.g., computing architecture, operating system, telecommunications needs) into which the product will be placed must be defined.
- If the entire system (hardware and software) is under development, system requirements must exist before software requirements elicitation should take place.
- Inputs to the requirements elicitation methods process include a vision statement, high-level business objectives for the project, statement of need, feasibility statement, statement of scope, and the project plan.
- Outputs from the requirements elicitation methods process include a list of prioritized requirements organized by function, domain constraints, a set of use case scenarios, and prototypes, if applicable.
- The rationale for each requirement should be recorded.
- Use case scenarios and mind mapping may be used alone, or as subtechniques under the interviewing, brainstorming, FAST, or JAD methods.
- With the appropriate people present, elicitation methods result in decisions.

With these common elements of requirements elicitation in mind, we'll move on to a discussion of some of the methods.

Interviews

In the chronicle of requirements gathering, interviewing is almost always used at some point. Even when the recommended brainstorming, mind mapping, FAST, and/or JAD techniques are used, it is usually necessary to understand the level of detail that follows concept exploration by interviewing at least some of the stakeholders.

In the past, analysts have simply made an appointment with representatives of the stakeholders, asked them what they want, and expected to gather all requirements in this fashion. Unfortunately, uncovering requirements using this method alone can be pretty tough. Among other issues, it is very difficult to meet face-to-face with enough stakeholders to represent all of the varied experiences and needs; in our global economy, they are likely to be located all over the world. Interviewing is, however, very useful as a first step in getting a handle on software requirements. It is widely believed that 50 percent of key intelligence information comes from human sources.

We recommend reading a few other works that concentrate solely on interviewing in general, such as *Basic Interviewing Techniques*, by Bruno Vanasse, or publications that specifically address software requirements elicitation, such as Gause and Weinberg's *Exploring Requirements: Quality Before Design*. There are some steps in interviewing that are so important and so basic that they can be presented here without contradicting other sources.

Steps in Interviewing

The steps in interviewing are: create the questions, select the interviewees, plan contacts, conduct the interview, close the meeting, and determine where to go from here.

Create the questions. Interviewing begins with the question, or set of questions, that will lead to the most knowledge about true requirements (what is really needed, not what is wanted or only thought to be needed). There should be some indicators from the original request for service, the concept exploration documents, and the software project management plan (SPMP) that can be used to create a template of interview questions.

Select the interviewees. Selected stakeholders will provide the human source for interviews, providing the most relevant and precise information than any other source. Most likely, it will be impossible to interview every stakeholder and representatives must be chosen. They should have know-how, accessibility, and the potential to credibly and reliably answer your questions.

There are several groups to consider in the hierarchy of interviewees:

> Entry-level personnel—Because they have not had the opportunity to gain much experience, they will not be able to contribute a great deal. However, they are worthy of an interview because of their fresh perspective and the possibility of an unexpected "pearl of information."
>
> Mid-level stakeholders—Based on experience, these stakeholders know the intricacies of the operational and/or technical side of the domain and can provide detailed insights. The project lead should always be interviewed.
>
> Managers or other special customers—CEOs and VPs who are knowledgeable of the domain and the impact of project success are definitely a stakeholder group to be interviewed. The executive sponsor should always be interviewed, if at all possible.

Academics—If there are such stakeholders, they can open your eyes to a different perspective.

Typical "users" of the system—This group is perhaps the most important because they will spend more time interacting with the system that anyone else. They may be revolutionary thinkers, which is good, but they may also be biased toward existing systems.

"Rising stars"—If they exist in the organization for which the software is under development, these soon-to-be-leaders in their field may provide cutting-edge thinking and information.

A multiplicity of human sources helps to verify the reliability of collected information.

Plan contacts. If time permits, do some research on the source you wish to contact and the perspective the individual has on the project—what is his "stake" as a stakeholder? Everyone is busy, so plan to limit the interview to what he wants to get out of the software; consideration of perspectives of other stakeholders will occur during brainstorming, FAST, or JAD sessions.

The first contact is often by telephone or email. Some guidelines are:

- Most people will be able to participate in a more relaxed interview on a Tuesday, Wednesday, or Thursday. They are likely to be less distracted by other work and in a better mood to talk.
- When calling, remember to use a pleasant tone of voice.
- Identify yourself clearly.
- State the purpose of your call or email.
- State how this stakeholder was chosen to be on the interview list.
- Confirm that he is the right person to answer questions and that the questions may be considered official.
- Give an estimated length of time for the interview.
- If available and needed, offer some compensation for travel and time given.
- Establish a time and place for the meeting, or a time for subsequent communication.
- Describe the exchange of information and the feedback mechanism.
- Ask for permission to study artifacts that portray the way the stakeholder currently does business.
- Describe the importance of a face-to-face encounter.
- If you encounter problems, leave your contact number in case the source changes their mind.

Conduct the interview. Interviews can be conducted over the phone, in person, or over the Internet (video conference, chat, email). However, the best way to perform interviews remains in the direct face-to-face human encounter. The most important factor for getting information in the interview is to get the right climate, one of comfort and confidence, between you and the interviewee.

1. **Establish the climate.** Once an atmosphere of trust and rapport has been established, it is possible to ask some very general and searching questions. The following actions will help in establishing that climate:
    - Ask for permission to take notes during the interview.
    - Exhibit a genuine human interest in the person's stake in the software.
    - Listen. The best motivator for an interviewee is your complete attention.
    - Start with smaller stuff and go to more sensitive issues.
    - Be direct and honest.
    - Be modest—nobody wants to talk to a "know-it-all."
    - Keep to the subject matter, unless you can tap on a shared experience that will contribute to understanding requirements needs.
    - Ask to see the environment in which the product will be used.
    - Offer future help or part of project report—make clear the exchange of information is a two-way street.
2. **Ask questions.**

    Keep them simple. Questions should be short and contain only one part. Two or three questions rolled into one can lead to compound requirements statements that are difficult to interpret and test.

    Bracket data. For example, "Does ABC Corp. sell between 15 and 30 million, or between 30 and 60 million dollars worth of widgets?"

    Keep an open mind. While it is important to prepare a set of questions, do not read from the questionnaire, nor only stick to it. There may be "golden nuggets" that can be discovered only by following the flow of the conversation, listening to the source, and branching into space that may be more pertinent than the prepared questions. Some of the interviewees may be experts in their field, so when they provide information it may be useful to ask them for an analysis—the "so what?" and "what if?" questions that provide additional insight.

    Ask context-free business, process, and product questions, such as these suggested by Gause and Weinberg:

    - Are there any problems this system could create?

- Are there reasons for wanting to solve this problem that might not be obvious?
- At what level of granularity would you like to see it?
- Describe the environment in which the system will operate.
- In what way will the system change the way you are doing things now?
- In what ways will the system help you to be more efficient?
- Is either data or functionality shared by other departments or business areas in the organization?
- Is there anyplace else the solution could be obtained?
- What current problems should this system solve?
- What data, needed by you, exists in other systems?
- What information will the new system deliver that you currently don't get?
- What is the most important business goal of the proposed system?
- What kind of performance issues do you have (if any)?
- What problems do you want this system to solve?
- What software application packages are you currently using?
- What will the new system accomplish (functionality) that is not currently achieved by an existing manual or automated system?
- Who is the client for the product?

3. **If the interview stalls ...**

   Perhaps the interviewee does not have, nor can have access to, the information you need. Once this is established, thank the person for his or her time and leave a good impression of your visit by leaving a contact number in case that person finds something for you.

   On the other hand, the interviewee may not feel like giving you the information. In this case, perhaps emphasizing his or her expertise and its positive impact on the system may get the interview past this hurdle.

   At times, the interview may need to be redirected by asking precise questions and giving examples of the type of answers needed or analysis required.

Close the meeting. Remember to re-motivate the person, because you never know when you could need her help again or when she may volunteer further information. Here are some suggestions:

- Ask if there are questions the interviewee has for you.
- Ask if there are other questions that should have been brought up.
- Leave a future objective, in the form of a future question or point of interest for you.
- Offer to share the requirements documents and Software Requirements Specifications (SRS).
- Leave a way to be contacted if more information is forthcoming.
- Ask if you can contact the interviewee if you think of another question.
- Ask if there is someone else that should be interviewed.
- Ask if there is hard-copy or electronic material to be taken and studied.
- Ask meta-questions about the effectiveness of the process—did the questions seem relevant?

Determine where to go from here. Do the following:

- Contact the source to thank him or her for participating.
- Document the interview findings.
- Prepare input for the next step of the process—brainstorming, FAST, JAD, or perhaps, go directly to the requirements specification.

Brainstorming Sessions

> No idea is so outlandish that it should not be considered with a searching but at the same time steady eye.
>
> —Winston Churchill
>
> Brainstorming is a conference technique by which a group attempts to find a solution for a specific problem by amassing all the ideas spontaneously by its members.
>
> It is easier to tone down a wild idea than to think up a new one.
>
> —Alex Osborn

The concept of brainstorming is explained in a body of knowledge, published in a myriad of books and articles (see the References section for a few). Here, we will simply summarize the suggested roles and rules of brainstorming as they apply to software project requirements elicitation. "Professional" requirements gatherers will want to study the works in the original, practice them under a master brainstorming facilitator, and contact a vendor for specialized training and automated tools. Others may find enough information in this chapter to proceed. An important point for the requirements elicitation process is that brainstorming is a great way to lead into JAD sessions and use case development sessions, as described later in this chapter.

Basically, brainstorming uses group effect to generate ideas and solve problems. It focuses on quickly generating ideas without evaluation or clarification. Anything goes—ideas may be wild, crazy, or impractical. Following idea generation, they are evaluated, and the ones to pursue are evaluated.

Definition of Brainstorming

Brainstorming is a group technique that may be used during the requirements-gathering process to promote creative thinking. It facilitates defining the "problem" to be solved by the life cycle activities to follow—including writing the software requirements specification, analyzing and modeling the requirements, and designing the software.

Brainstorming is used in many business applications as a way of generating a lot of new ideas, quickly. In this particular business situation, we are trying to brainstorm the needs of stakeholders in order to avoid unpleasant surprises later in the project. The more requirements that can be generated up front, the better. Even if they can't all be delivered at once, or in the first version, or ever, there is an advantage to having them documented. If a new generation of users requests a feature that has been thought out and postponed or deleted, time and effort will be saved since the interviewing, brainstorming, FAST, or JAD records document when the idea was first proposed and why it was not implemented.

A group technique, brainstorming is intended to generate lots of ideas, with the full understanding that they may not all be useful. The theory is that having a long list of requirements from which to choose is far superior to starting with a blank slate. Items in the long list can be categorized, prioritized, and pruned.

Alex Osborn is given credit for the central concept of brainstorming.[11] In 1941, as an advertising executive, he was looking for ways to stimulate new ideas, instead of inhibiting them, in formal meetings. He said groups could "think up" ideas by using a group process where all ideas are welcomed—none are criticized, ideas are allowed to build upon each other, wild and exaggerated ideas are encouraged, and large quantities of ideas are produced. His theory was that a greater quantity of original ideas gave rise to a greater quantity of useful ideas. Individuals alone will seldom, if ever, produce the volume or creativity of ideas made by a small group of individuals engaged in brainstorming techniques.

Osborn found that when natural inhibitions are reduced, people are far more willing to put forth useful ideas that they might otherwise feel were of no value. In fact, sometimes the more "far out" or zany the idea is, the more it changes the way others think—a way to get started with what we now call "thinking out of the box."

The development of this original technique was revolutionary—brainstorming is now used by nearly all of the world's largest companies and has become synonymous with "creative thinking" in most vocabularies.

A brainstorming session entails a group of people who free themselves from social inhibitions to generate as many ideas as possible so that original thoughts are free to surface. All participants are encouraged to say whatever ideas come to mind, whether they seem relevant or not. No one will be criticized for any idea, no matter how goofy it seems, as the responsibility of the participant is to generate views, not to vet them.

Dictionary definitions of "brainstorm" include: a sudden inspiration; a bright idea; a severe outburst of excitement, often as a result of a transitory disturbance of cerebral activity; a sudden mental aberration. Various modern authors have characterized the brainstorming process as:

- a part of problem solving that involves the creation of new ideas *by suspending judgment;*
- a technique that maximizes the ability to generate new ideas;
- a time dedicated to generating a large number of ideas regardless of their initial worth;
- designed to obtain the maximum number of ideas relating to a specific area of interest;
- the creation of an optimal state of mind for generating new ideas;
- the free association of different ideas to form new ideas and concepts;
- where a group of people can put social inhibitions and rules aside with the aim of generating new ideas and solutions.

As we will see, the definitions give way to "rules" to be followed in a brainstorming session.

Roles for a Brainstorming Session

There are three roles for participants in a brainstorming session: leader, scribe, and team member.

Leader

The leader is also called the facilitator, or moderator. Trained to be a good listener, the leader is responsible for making the brainstorming process an easy one, and the session itself run smoothly. He or she will encourage the participants, ensure proper individual and group behavior, and assist the scribe in capturing ideas. The leader will follow a published agenda and restart the creative process if it falters. The leader will ask for ideas to be shouted out, but will also accept written ones.

Scribe

The scribe will record every idea in such a way that everyone in the room can see it. Flip charts and markers, white boards, or overhead transparencies work for this, as does the projection of a computer screen. If using flip charts, remove each filled piece of paper and tape it to the wall so that everyone can see it. In a small group, the leader could also be the scribe, but it is difficult to keep the meeting moving *and* capture each idea correctly. In a large group, two or three scribes may be necessary.

Participants

In software development there are usually several stakeholders representing varying points of view. Brainstorming sessions are a wonderful venue for them to learn to empathize with each others' points of view. The good news is that differing personalities and vantage points will result in more creativity and a broader outlook. The risk is if the participants can't stop protecting their own needs for the sake of the process. Group sizes can number between four and 30 people, but seasoned leaders have recommended groups of five to 10 (the ideal group is six or seven). More people allows for the opportunity for diversity, but can lead to nervousness or frustration if there is not time to express all ideas. The primary responsibility of participants is to produce ideas and thoughts (requirements in this case), and to stimulate thoughts in others.

Rules for a Brainstorming Session

The rules for a brainstorming session have been published in many ways, with each often overlapping the other. We'll optimize our time here by consolidating rules from multiple publications into one comprehensive list.

Rules for Leaders

- Allow ideas to be expressed verbally (preferred) or in a written note.
- Allow silence when appropriate.
- Collect as many ideas as possible from all participants, with no criticisms or judgments made while ideas are being generated.
- Communicate the ground rules of the session.
- Don't allow criticism, discussion, or debate.
- Don't allow serious or derisive commentary.
- Encourage wild or exaggerated ideas.
- Establish the purpose or topic.
- For the most part, keep a fast pace to reduce inhibitions and evaluation.
- Help the participants mutate and combine ideas.
- Help the participants with the generation of ideas.
- Set a time limit.
- Use all the time allotted.

Rules for Scribes

- Don't describe an idea in detail—just capture its essence.
- If necessary, ask for a brief clarification.
- Use the speaker's own words.
- Write down each idea as stated, in short words or phrases.
- Write ideas so that all participants can see them.

Rules for Participants

- "Hitchhike," "piggyback," or build, on the ideas of others.
- Allow yourself to "be in the moment"—become absorbed by the process and think freely.
- Assume different personalities.
- Be creative in contributions.
- Be open to new, original ideas.
- Be patient with silence.
- Be willing to take risks.
- Build and expand on the ideas of others.
- Contribute only one idea at a time.
- Create improvements and variations of recorded ideas.
- Don't bother with providing explanations.
- Don't evaluate the ideas of others; don't criticize or judge.
- Don't give nonverbal clues that could be construed as criticism or judgment of others' ideas.
- Don't interrupt others.
- Don't place too much importance on any one idea.
- Generate as many ideas as possible.
- Generate as many ideas as possible in a short period of time—quantity now, quality later.
- Keep each idea short.
- Let your imagination soar.
- Reflect later.
- Take turns making contributions.
- Think fast.
- Write down your idea and pass it to the scribe.

Brainstorming Rules in General

- All ideas are welcome—there is no such thing as a "wrong" idea, or a silly one, or a dumb one, or one that is too far out.
- Each idea presented belongs to the group, not to the person who said it.
- Every point of view is valuable.
- Have the right number of people in the session.
- Have the right people in the session.
- Talking about the ideas will take place after brainstorming is complete.

An Expansion of Some of the Most Important Brainstorming Rules

Postpone and withhold judgment of ideas. Ideas can be discussed following the brainstorming session, but no judgment may be passed during the meeting. No one should suggest that an idea might not work or could have negative side effects. All ideas are potentially good (even if they don't seem so, they may be modified to produce great benefit); any discussion is likely to evolve into criticism or compliments. The evaluation of ideas takes up valuable time and effort that should be devoted to the creation of ideas instead.

Encourage wild and exaggerated ideas. It's easier to tame a wild idea than it is to think up a new and workable one. Wacky and seemingly unworkable ideas may inspire valid ones by stimulating new thought patterns. No idea is too bizarre or nonsensical—producing them will reduce inhibitions in others as well.

Quantity over quality, for now. It is easier to pare down a long list of ideas, or to create one good idea from combining lots of little ideas, than to try to supplement a puny one. The chances of finding a good idea improve with a long list. The leader should consistently help extract as many ideas from the participants as is possible during the course of the session.

"Hitchhike" on ideas. Most creative people are also good listeners. Leaders should encourage taking recorded ideas and expanding upon them, adding extra thoughts, adapting and improving them, using them as inspiration for original ideas, or for combinations of ideas. Each idea has a principle or concept that is useful.

All participants and all ideas have equal worth. Viewpoints can and should be different, but they are all valid. Unique perspectives on situations bring a richness of solutions.

Suggestions for a Brainstorming Session

In addition to roles and rules for a brainstorming session, experience has shown that a number of subtle techniques will add to the chances for success:

- Allow the group to determine the actual length of short breaks, as the freedom to start and stop when they are ready is a relaxing ingredient. Refreshments are helpful.
- At the opening of the brainstorming session, have a brief warm-up on an unrelated but fun topic. This will help set the mood and increase the comfort level of those who have not attended such sessions before. When the main session begins, the creative juices will be flowing and an unrestrictive mood will prevail.

- Be sensitive to the fact that many employees fear making mistakes in their jobs—fear of their managers, of losing their job, of demotion, of loss of status—and they are being asked to submit wild ideas that may not work. Creating a safe environment for them is a crucial part of allowing their creativity to surface. If properly set up, the brainstorming session is ideal because all participants have agreed not to judge. "Crazy" ideas are actually encouraged.

- Consider the concept of mind mapping to augment brainstorming, as discussed in the next section of this chapter.

- Don't trust first impressions that may make ideas seem unappealing; consider "seeds" of ideas.

- If the flow of ideas seems to be drying up, ask small groups to congregate around different flip charts and discuss the ideas on them. Another technique is to ask each person to write some ideas on a piece of paper and exchange it with someone else to discuss and build upon. List alternatives; "stuck" thinking doesn't mean "stopped" thinking.

- It works well to have the participants' seats arranged in a circle with no "head" of the table. U-shaped tables are also effective.

- Look for underlying concepts behind common things; use the process of "association."

- Make refreshments available during the entire session, limiting the time lost to breaks. A "stretch" period should still occur every few minutes.

- Place flip charts and markers within easy reach of each participant. Supplying the participants with notepads and pens facilitates the notation of personal ideas during the active moments, ensuring that no idea gets lost.

- Provide a set of things for the participants to manipulate with their hands to stimulate the thought process. Crayons with paper, and colored children's modeling clay are two good examples.

- Provide the participants with comfortable chairs.

- Provide visual variety in the room, giving participants something to look at while someone is suggesting an idea—it can be uncomfortable for either or both if everyone is looking the contributor in the face.

- Realize that if anyone needs a microphone to be heard, the group is too large for brainstorming.

- Reserve a room that is large enough to provide space for the participants to move around with comfort, but not so large that it seems cavernous and makes the group feel small in comparison.

- Restrict brainstorming sessions to two hours or less; break the period into 15-minute-or-less segments.

- Return to the posted ideas for reflection and perhaps combination or expansion, as the brainstorming session traverses peaks and valleys of rapid idea generation.

- Seat very small groups at a table covered in butcher paper for free-form writing. Participants should not be seated at too great a distance from each other.
- The leader may become part of the group and play an equal role in idea generation.
- Upon restarting the session following a break, ask people to sit in a different place and meet their new neighbors before starting again.
- When addressing the group, use "we" instead of calling participants by their names. This will enhance group bonding and reinforce the concepts of group effort and group responsibility.

Brainstorming Session Agenda

The following is a proposed agenda for a brainstorming session, consolidated from a number of sources (see the References section):

1. Introduce the session. Review the topic of the brainstorming session, discuss the rules, and introduce the participants.
2. Conduct a warm-up. Provide a warm-up activity (five to 10 minutes) that helps the group get used to the feelings of a safe environment, as well as the excitement that can be mounted through idea generation. The subject of the warm-up is neutral, something understood by all (for example, ideas for new kitchen gadgets or automobile accessories), and not related to the topic at hand in any way. In addition to fostering creativity with the warm-up subject, the leader could discuss the rules and ask the group to suggest additions or changes to make the session a particularly effective one.
3. Brainstorm. Ask for reasonable ideas, radical ideas, ideas that might work in a strange way, and ideas that just spring to mind. Work for 10 to 15 minutes and call time. Stop before all of the excitement wanes. If the group is in the "white heat of inspiration," allow them five more minutes. Take a break and then resume, always following the rules. Neither force a break nor force a return from a break. If ideas appear to be drying up, process the ideas and review them for clarity; ensure that every participant understands them. Combine and categorize similar ideas and delete duplicates. Determine the criteria for evaluation.
4. Establish a consensus. Ask the group to vote on a small of number ideas to consider first (10 to 15, or about one-third of the refined idea list). Give each participant the same number of points to allocate to the idea list. Allow time for thinking about the "voting," then ask each participant to anonymously submit their allocations.
5. End the session. Thank the participants for taking part, allow them to finish if they are in the middle of writing, ask them to fill out an evaluation form, and tell them when the collated ideas will be fed back to them. Invite latent ideas.

Preparation for Brainstorming

As with projects, brainstorming sessions that are planned and prepped will have a much greater chance for success. The following are "must do's":

- Identify and invite the participants, the scribe, and the session leader; distribute information about the location, time, place, expected length of session, and RSVP-by-date.

- Identify the specific software system or subsystem to be developed and publish appropriate documentation (e.g., concept exploration summaries, interview summaries, project planning objectives) to all group members.

- Define brainstorming and brainstorming session rules and roles for your organization and distribute them to the participants.

- Ensure the room is properly equipped with comfortable chairs, appropriately shaped or arranged tables, flip charts, white boards, Post-its, projected computer monitors, refreshments, and manipulable "toys" (clay, crayons, kaleidoscope, etc.).

Follow-up to a Brainstorming Session

Typically, it is the leader who edits the ideas after a session. We recommend that he or she involve another person or two who is familiar with the application under development.

First, discard ideas that are unusable or incomplete (don't throw them away—document them for the record, but remove them from the list of ideas under consideration). Use valid criteria for removal of ideas and take care not to remove any too early. Remove duplicates if any still remain. If the session team had time to provide their short list of unanimously recommended ideas, this step is already completed. All of the ideas should be in one sectioned list, in some computerized format (e.g., an Excel worksheet).

Arrange the remaining ideas into three categories:

Valid. Appears to be a needed, testable requirement.

Possible. May be a requirement.

Not Likely. Probably is not a valid requirement.

Document the findings for input to the software requirements specification. They will not only help in turning requirements into specifications, they will help in determining which specifications will be delivered in the first implementation of the product. The prioritization process will be discussed later in this chapter.

Mind Mapping

Mind mapping is a concept that may be used in conjunction with brainstorming, or alone. The idea is that the use of keywords, along with pictures, symbols, and color can capture the way our brain perceives subject matter. For many people, particularly those whose learning style is visual, recall of details is easier in this nonlinear language style, making it particularly useful for quick summarization. Mind map advocates contend that 90 percent of written words are not useful in the task of recalling content, but that this "right-brained" activity provides highly effective visual tips. The development of this basic technique is attributed to Tony Buzan and Joseph Novak.[12]

The following is a brief history and some combined definitions:

- A mind map is a drawing, or sketch, of central and associated ideas that are represented by pictures (icons, graphics), and keywords that are written on lines (arcs, vectors) attached to the pictures.

- Every mind map is unique in its form and content. This is especially relevant for the task of recalling information.
- Keywords and connections between the keywords are easily seen due to the lines connecting them.
- Mind mapping makes possible the transformation of complex net-like fields of knowledge into a manageable structure.
- The central idea is presented clearly by its central location on the map (drawing); the relative importance of each idea becomes obvious by its placement—ideas of greater importance are closer to the center, those of less importance are closer to the boundaries.
- The idea of mind maps is to integrate a range of information concerning the central topic into a wholistic structure.
- The transparent and essentially wide-open mind mapping scheme facilitates the generation of new ideas in the creative aspects of note-taking.
- These constructs allow the informational content to be memorized and recalled with greater ease and increased efficiency.
- This type of structure makes it possible to integrate new information easily and without compromising the readability of the structure through deletions and cramped insertions.

In the construction of mind maps, the requirements elicitor will:

1. Select memorable keywords.
2. Connect keywords by writing them on lines and connecting the lines.
3. Structure the content of the mind map differently from the linear structure of the written page. Instead of beginning at the top of the page and proceeding vertically, sentence by sentence, a mind map begins in the center of the drawing and radiates outward from the main topic.

Group-based mind mapping, such as the activities that accompany brainstorming, involve the drawing of a group-created set of central ideas that are mounted on the wall in an "art gallery" of ideas. Individuals then visit each drawing and add inspired, connected words and sketches. Eventually, the subject will be documented in a way that allows quick and easy recall of the issues/solutions/requirements.

Facilitated Application Specification Techniques (FAST)

FAST is similar enough to brainstorming and to JAD that we will provide only cursory coverage here. This is not to belittle the process in any way—FAST is an important part of the software PM's vocabulary because it was developed to be a method specifically for gathering software requirements. But rather than attempt to rewrite existing texts, we will only summarize FAST and refer you to other works, such as Pressman's *Software Engineering*.[13]

The general rules for FAST will seem familiar, as they are similar to those for brainstorming:

- Conduct the meeting at a neutral site, attended by developers and customers.
- Establish rules for preparation and participation.
- Publish an informal agenda.
- Invite a facilitator to control the meeting.
- Prepare a definition mechanism—worksheets, flip charts, wall stickies, and so on.
- Participants should agree not to critique or debate.
- Share a goal:
    1. Identify the problem.
    2. Propose elements of the solution.
    3. Negotiate different approaches.
    4. Specify a preliminary set of solution requirements.

Preparation for a FAST session might include:

- making a list of objects that are:
    1. part of the environment that surrounds the system;
    2. produced by the system;
    3. used by the system.
- making a list of services (processes or functions) that manipulate or interact with the objects.
- making a list of constraints and performance criteria.

Activities during a FAST session typically include:

- presentation of prepared lists of objects, services, constraints, and performance for discussion (as separate entries so they can be moved, combined, etc.).
- creation of a combined list, where redundant entries are eliminated, and new ideas are added.
- consensus on the list, gained from the participants by the facilitator.
- work, by teams, on "mini-specifications," or discrete portions of the known software requirements:

1. discuss mini-specifications;
2. identify externally observable data objects;
3. evaluate the flow and content of information;
4. define and elaborate software functionality;
5. understand the software "behavior";
6. establish interface characteristics;
7. uncover design constraints.

- maintenance of a "parking lot list of issues" to be resolved later.
- creation of validation criteria.
- assignments for the next step of drafting the SRS (see Chapter 17).

Joint Application Design

In addition to one-on-one interviews, interviews in small groups, brainstorming, and FAST sessions (with or without mind mapping), another type of extremely effective facilitated meeting is the Joint Application Design (JAD). As with the other requirements elicitation methods, it has been described in numerous other works. Wood and Silver wrote the definitive book in 1989, which remains popular today.[14] Here, we will present an overview of the JAD process; Appendix G, "Joint Application Design in Use," offers a glimpse into a recently completed real-life JAD exercise held between two major, just-merged, U.S. companies.

Definition of JAD

Joint Application Design (JAD) is a registered trademark of IBM Corporation. It is a team approach to requirements elicitation, focusing on improving the group process and getting the right people involved from the beginning of a project. Toby Crawford and Chuck Morris at IBM developed the initial concept of JAD in 1977. Through a marriage of the IBM methodology, business systems planning, and their own innovative methods, they got IT professionals and end-users to agree on requirement and design specifications. Since 1977, JAD has emerged as the most widely used tool for requirements definition. It isn't just for requirements, though—it is employed in the creation of other plans, designs, and policies throughout the software life cycle. Development organizations now realize that a methodology with a high degree of user interaction (including prototyping and RAD) leads to higher-quality software.

JAD sessions are similar to brainstorming sessions, but they aren't quite the same animal. Brainstorming sessions last about two hours; JAD sessions tend to last about three days. Brainstorming sessions are about rapid-fire production of ideas; JAD sessions can produce high-level, but specific, software models of function, data, and behavior.

JAD is a structured and disciplined session led by a facilitator. It is based upon communication through documentation, fixed requirements, and rules of work. As the JAD methodology evolved, it began to use CASE and other software tools during the sessions to capture work products such as data flow diagrams (DFDs), entity relationship diagrams (ERDs), state transition diagrams, and all

commonly used object-oriented diagrams.

A 1989 study by Capers Jones of 60 development projects found that 35 percent of the functionality was missed without JAD, while less than 10 percent of the functionality was missed when JAD was incorporated into the requirements elicitation process. Projects combining JAD and prototyping did even better at managing requirements refinement.[15]

Roles in JAD

Developers

The role of the developer is to assist stakeholders to formulate their needs, which are usually solutions to existing problems. The stakeholders share ownership of the software requirements.

Participants

Like the participant roles in brainstorming, these invited stakeholders are the most important part of this process. The knowledge and skill level of the invitees are crucial to the session's success. Having the right people allows decisions to be made rapidly and models to be developed that are correct, even if incomplete.

The Facilitator/Process Consultant

As with brainstorming, the facilitator keeps the session focused and flowing, and prevents or minimizes the acting out of unproductive human emotions such as attacks or defensiveness. The facilitator owns neither the process nor the product, but is present to help the stakeholders with their product creation process.

Scribe

As with brainstorming sessions, the JAD scribe(s) documents ideas and helps with timekeeping.

JAD Sessions

According to Wood, JAD provides a concentrated workshop for making decisions with everyone present who has the authority to make those decisions. She recommends phases of project definition and research to deal with fact-finding and information gathering, preparation for the JAD session, and the JAD session itself, to validate the gathered information. As with brainstorming, the JAD session will be supported with a properly outfitted room, including access to white boards, flip charts, markers, appropriately shaped and spaced worktables, projectors, wall charts, refreshment tables, and so on.

Ground rules/guidelines for JAD sessions mirror those for brainstorming sessions:

- There are no "dumb" questions.
- All participants are equal.
- Silence means agreement, even (especially) if you're absent.
- Any team member can request to go around the room and get explicit agreement from each person on a given issue.

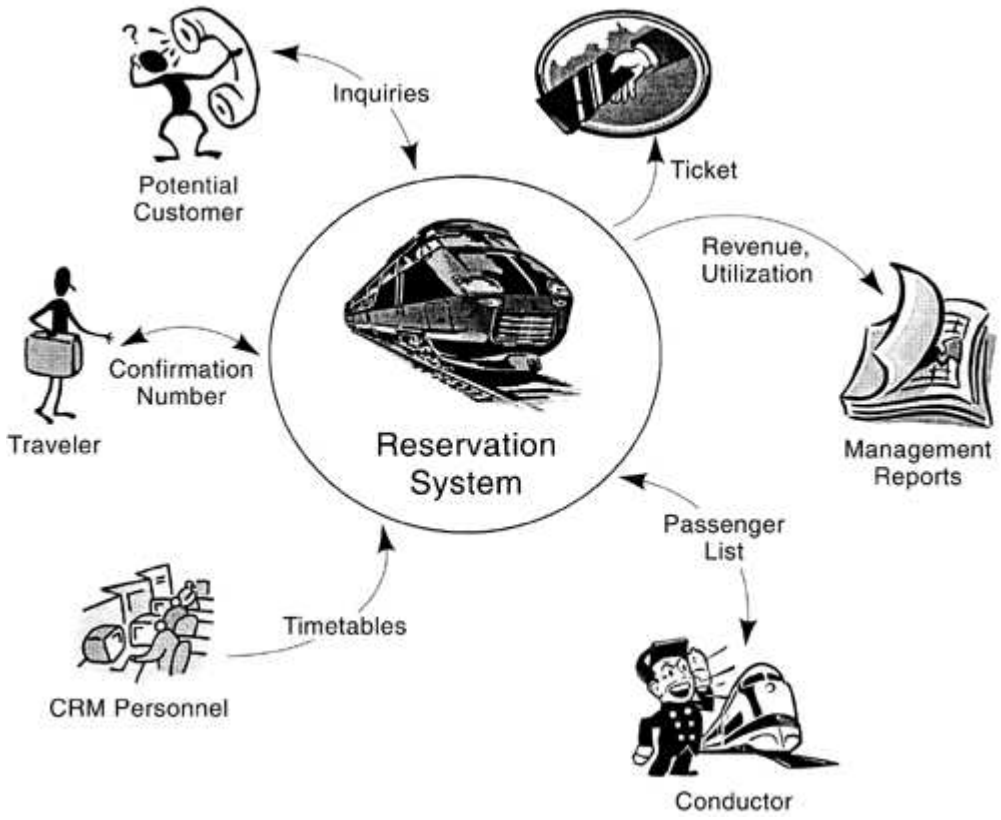
- No "speeches" are given.
- No side conversations are allowed.
- All team members are involved for the full period of time (three days are recommended).
- One user scenario will be addressed at a time.
- Brainstorming is for quantity, not quality.
- Define terms and capture three-letter acronyms (TLAs).
- All team members participate with an open mind.
- All team members support each other, learn, grow, and relax and have a little fun.

Output from a JAD Session

It makes sense that the two groups, developers and stakeholders, will jointly own all deliverables from the JAD session where the joint authorship occurred. Frequently, these software-specific artifacts will include:

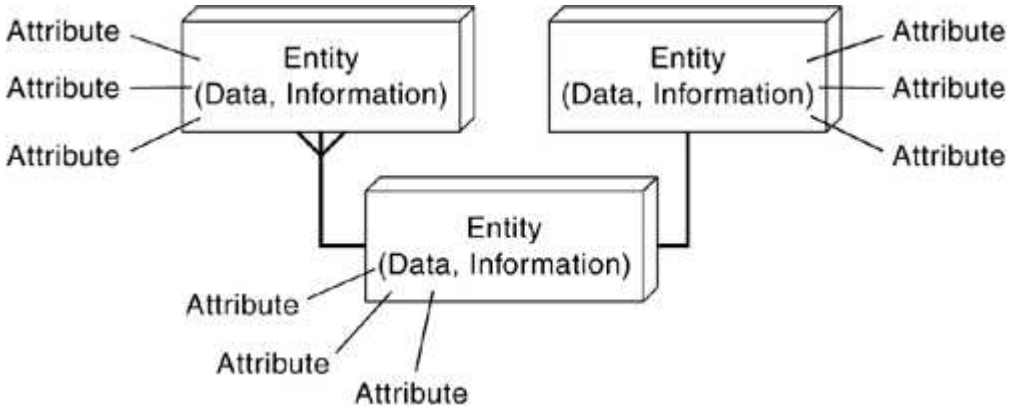
- a data context diagram (see Figure 16-6);

Figure 16-6. Example of a Partial Data Context Diagram (DCD)



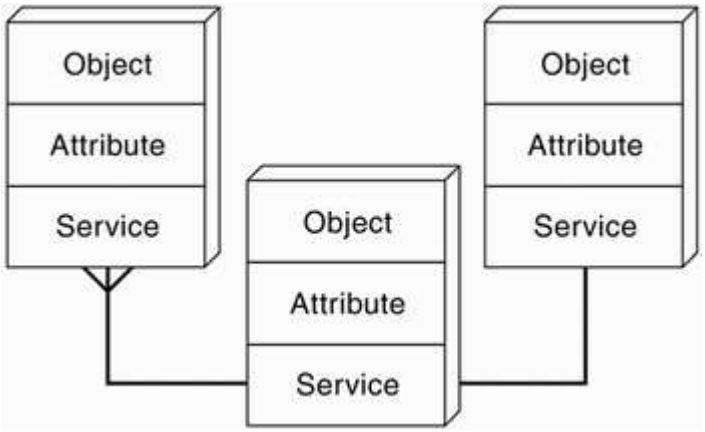
- a Level 0 data flow diagram;
- a Level 1 data flow diagram;
- a global data model—entity relationship diagram (see Figure 16-7);

Figure 16-7. Example of a Global Data Model



- a list of primary objects;
- a high-level object model (see Figure 16-8);

Figure 16-8. Example of a High-Level Object Model



- candidate responsibilities and collaborators for each object;
- a list of primary processes/use cases;
- other data flow diagrams, state diagrams, decision trees, or decision tables where needed;
- data requirements for each process;
- a list of assumptions;

- documentation for the resolution or assignment of open issues.

Output from a JAD session used to elicit requirements paves the way for the next step—the creation of the SRS.

Possible Issues with JAD

No method is perfect, and while the JAD technique has been used successfully for several decades, there are a few recognized issues.

For example, participants funnel their ideas through a facilitator and/or a scribe. Precautions must be taken to avoid misinterpretation of the collected data. The use of automated tools during the session and the review of the session deliverables by all participants reduces this risk.

JAD sessions have dealt primarily with IT systems, which emphasize data elements and screen design. The success of JAD used with real-time systems requirements elicitation has not been well documented.

A three-day JAD session with representatives of all stakeholder groups, all having decision-making authority, is an expensive proposition. Three days is only an average—a complex embedded real-time system, or a system upon which human life is dependent, could easily take longer. If the length of the session is "until we are done," a measure of "doneness" could be that no more use cases can be identified.

User Scenario and Use Case Development Sessions

As with so many other topics in this survey of practical methods, all things object-oriented constitute a huge body of knowledge. And, as we indicate with almost all of the subjects in this book, we are barely able to scratch the surface. We describe the basics of what a software project manager needs to know in terms of balanced, useful project, product, and people skills and give references for digging deeper into each area. Here, we have space to briefly mention use cases as a requirements elicitation technique. Object models will be addressed in Chapter 22, "Analysis and Design Methods."

Requirements elicitation is a combination of textual and graphical models used to represent the intended system, detect inconsistencies, errors, omissions, and ambiguities. The text and pictures improve understanding of the requirements, bridging gaps in language and vocabulary. There are many software project managers who believe strongly that creating use case scenarios and diagrams during requirements gathering sessions (JAD, FAST) is a key to definition, user-system interaction, and a basis for validation.

Definition of a Use Case

The use case model is about describing what (not how) our system will do at a high level and with a user focus for the purpose of scoping the project and giving the application some structure. Use cases are scenarios that identify a particular thread of system usage—a written narrative. The terms *use case*, *use case scenario*, and *use case diagram* are often interchanged, but in fact they are different. Use cases are structured outlines or templates for the description of user requirements, modeled in a structured language (English). Use case scenarios are unstructured descriptions of user requirements. Neither is limited to object-oriented methods, but that is the arena in which they are usually discussed. Use case diagrams are graphical representations that may be decomposed into further levels of abstraction. In order to describe them, their components will be described first.

Actor

An *actor*, or external agent, lies outside of the system model, but interacts with it in some way. An actor can be a person, machine, or an information system that is external to the system model; an actor is not part of the system itself. Customers, client application, external systems (e.g., legacy systems, accounting systems), and external devices (e.g., fault monitors) could all be actors.

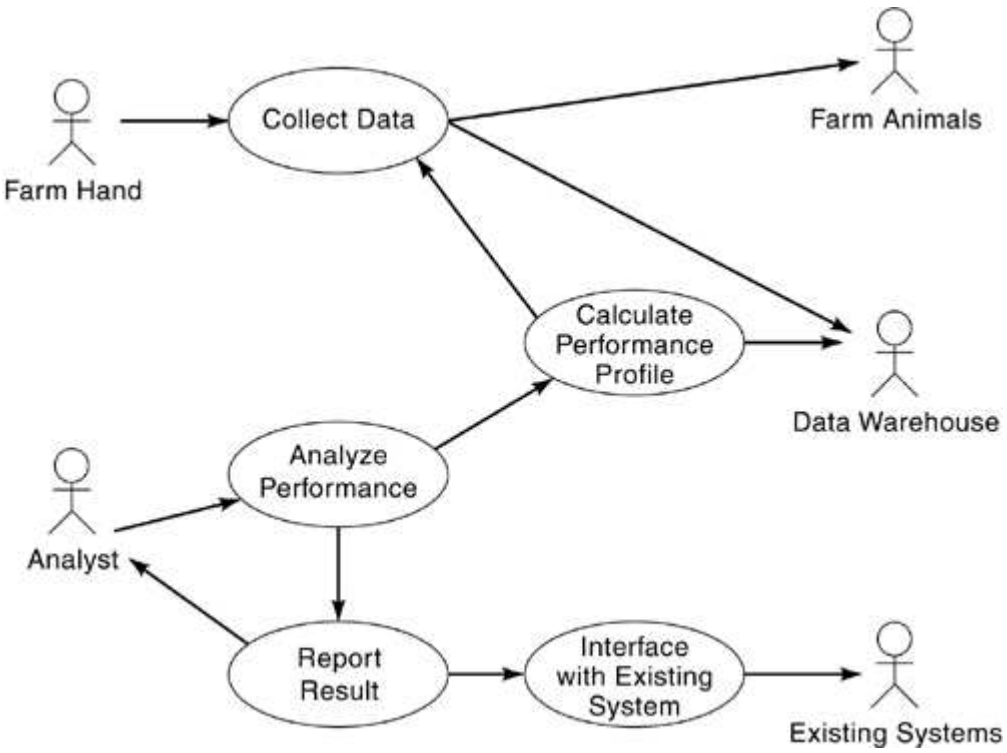
Use Case

A *use case* is narrative text describing the sequence of events of an actor, using a system. They are "stories" of how a system is used, illustrating and implying requirements. They are transactions (atomic sets of activities) that yield identifiable value to the actor. They represent a description of the courses of events that will be carried out by the system. A use case is a static model of functionality, having static relationships, implying no temporal sequencing.

Use Case Diagram

A *use case diagram* visually represents what happens when an actor interacts with a system. The system is shown as a rectangle with the name of the system (or subsystem) inside; the actors are shown as stick figures (even the nonhuman ones), the use cases are shown as solid-bordered ovals labeled with the name of the use case, and relationships are lines or arrows between actors and use cases and/or between the use cases themselves. Actors appear outside of the rectangle since they are external to the system. Use cases appear within the rectangle, providing functionality. A relationship or association is a solid line between an actor and each use case in which the actor participates—the involvement can be any kind, not necessarily one of the actor initiating the use case functionality. Figure 16-9 shows an example of a use case diagram.

Figure 16-9. Use Case Example



Richter tells us that the use case models system functions by describing how an actor uses the

system. It represents a function that would make sense to an actor. Ask what interactions with the system the actor initiates, and what interactions with the actor the system initiates. A paragraph or more of text describes each use case, also known as a specification. First identify the system's actors and use cases, then combine the information into a use case diagram—a static model of functionality with static relationships, implying no temporal sequencing. It contains no information about the dynamics of these functions. Activity diagrams, which are described in Richter's book, *Designing Flexible Object-Oriented Systems with UML*, provide a solution to these deficiencies, depicting a workflow view of activities.[16]

Benefits of Use Case

The use case model can be a key tool in understanding the requirements of the business, as well as a way of ensuring that everyone who uses the system or works with it can agree on how the model will look. The following list identifies some of the benefits:

- Having consistent names for the different use cases facilitates communication and avoids errors caused by different people, or groups of people, thinking they are referring to the same thing when they really mean something different.
- Use case diagrams can provide a documented model that accurately describes the existing business, as well as the system under development.
- Use case models are understandable to managers, users, and developers, with no need for specialized training.
- Use cases and scenarios can provide validation as well as implementation testing.
- Use cases can be used to produce business rules.
- Use cases define the process by which the customer interacts with the group.
- Use cases may be easily transformed into object models.
- Use cases provide traceability for functions.
- Use cases provide the basic information needed for creating prototypes.
- Use cases reduce the occurrence of the "orphan function"—one that sounds good but is not a direct contributor to the system goals.
- Use cases show the courses of events that can be performed.

Guidelines for Creating Use Cases

As use cases are developed during requirements elicitation sessions, a few guidelines may be helpful:

- Allow each use case to tell a story about how the user interacts with the system under development to satisfy a discrete goal (the conceptual dialog).

- Assign project champions who are actual users for each user class to serve as the primary interface between users and developers.
- Avoid design strategies at this point, including user interface design.
- Avoid duplication across use cases.
- Avoid creating too many use cases too quickly.
- Don't record data definitions in use cases (data attributes belong in class diagrams, logical data models, and the project data dictionary).
- Record the following about each use case:
    1. a unique ID
    2. a concise name
    3. history—the originator, creation date, updater, last update date
    4. the actor(s)
    5. a brief description
    6. pre- and post-conditions
    7. priority
    8. frequency of use
    9. a detailed description of the user actions and system responses under normal, expected conditions
    10. alternative courses
    11. error conditions and system response
    12. includes (other use cases that may be called)
    13. special requirements
    14. assumptions
    15. comments
- There should be more functional requirements than use cases.

During an elicitation session like FAST or JAD, the session leader may use the following steps to get the use case definition process started (see Figure 16-10):

1. Identify who will use the system directly, for example, hitting keys on the keyboard—these

are the actors.

2. Pick one of those actors.

3. Define what that actor wants to do with the system; each of these major activities become a use case.

4. For each use case determine the "basic course," what normally happens. Describe that basic course in the description for the use case.

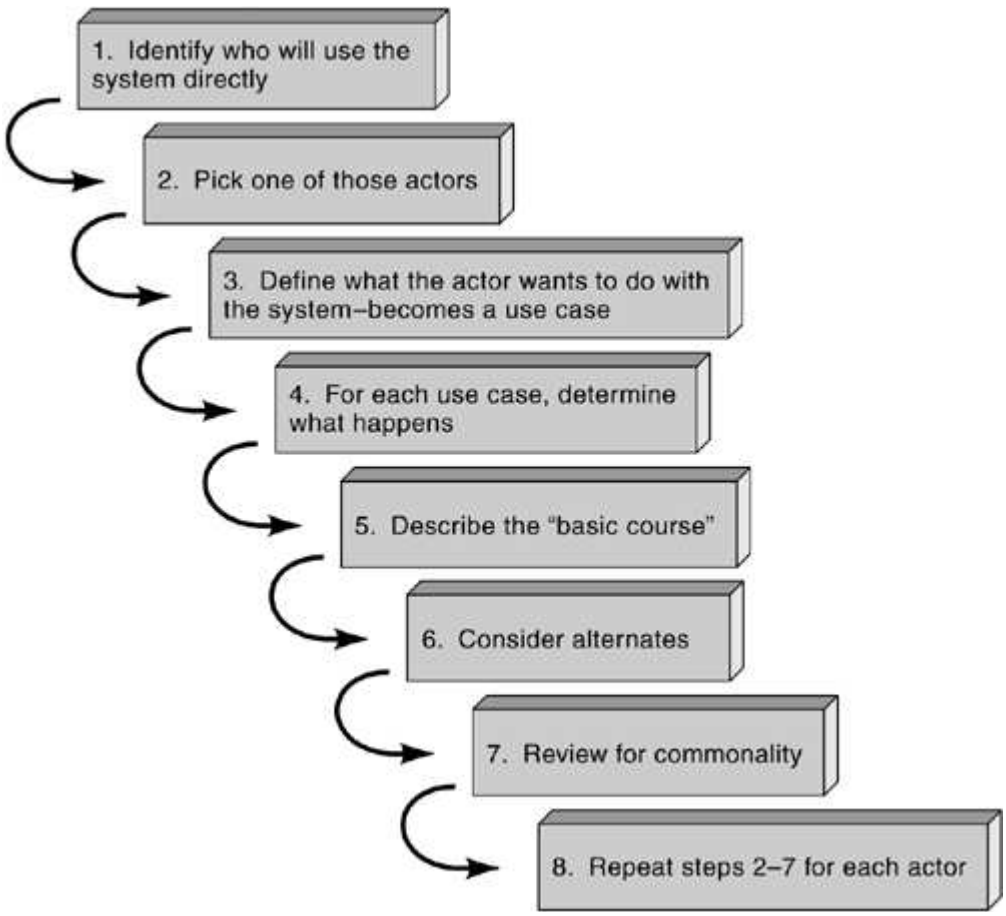
5. Describe it as "Actor does something, system does something; actor does something, system does something," but keep it at a high level. This is not the time to discuss user interface design. Describe only system functions that the actor would be aware of and what the actor does that the system would be aware of.

6. Once you're happy with the basic course, consider the alternates (they will be added as extending use cases, described thoroughly in all of the referenced object-oriented literature).

7. Review each use case description against the descriptions of the other use cases. Where commonality exists, document only one "used" use case to avoid duplication.

8. Repeat Steps 2 through 7 for each actor.

Figure 16-10. Steps in Creating a Use Case



The last piece of advice we will leave for the use case modeler is to "keep it simple."

PREVIOUS < Free Open Study > NEXT