2. REQUIREMENTS DEVELOPMENT
CH3. SOFTWARE REQUIREMENTS ELICITATION

SOFTWARE REQUIREMENT ENGINEERING - SRE (UNDERGRADUATE)

BUSINESS REQUIREMENTS: PRODUCT VISION AND PROJECT SCOPE

- "Business requirements" refers to a set of information that describes a need that leads to one or more projects to deliver a solution and the desired ultimate business outcomes.
- Business opportunities, business objectives, success metrics, and a vision statement make up the business requirements.
- Organizations should not initiate any project without a clear understanding of the value it will add to the business.
- Two core elements of the business requirements
- The product vision briefly describes the ultimate product that will achieve the business objectives.
  - The vision describes what the product is about and what it ultimately could become.
- The project scope identifies what portion of the ultimate product vision the current project or development iteration will address.
  - The statement of scope draws the boundary between what's in and what's out for this project.

PRODUCT VISION AND PROJECT SCOPE

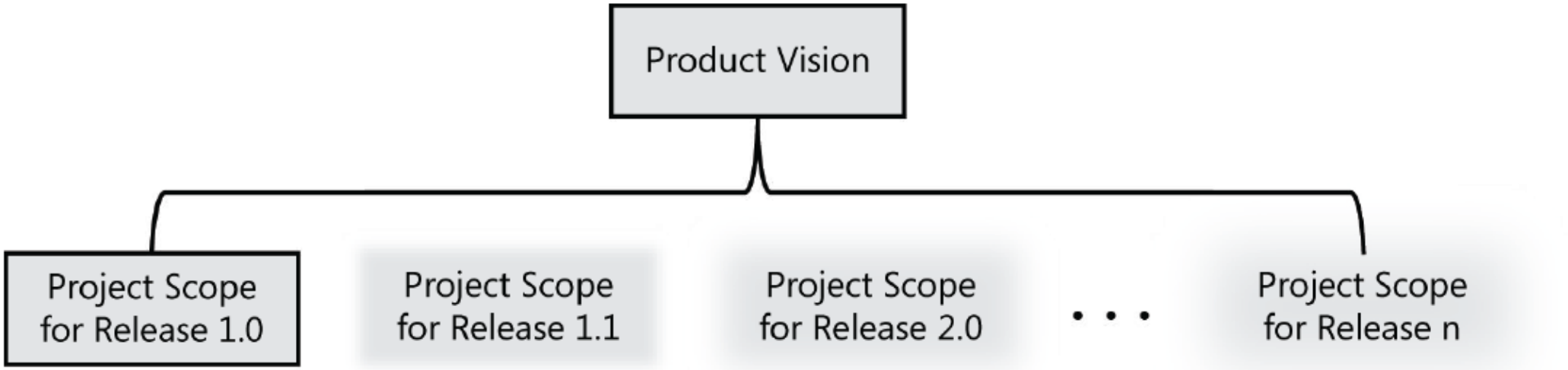

FIGURE 5-1 The product vision encompasses the scope for each planned release, which is less well defined the farther out you look.

VISION AND SCOPE DOCUMENT

- The vision and scope document collects the business requirements into a single deliverable that sets the stage for the subsequent development work.
  - Some organizations create **a project charter** (agreement) or **a business case document** that serves a similar purpose.
  - Organizations that build commercial software often create **a market (or marketing) requirements document (MRD).**

- The owner of the vision and scope document is
  - the project's executive sponsor, funding authority, or
  - someone in a similar role.

VISION AND SCOPE DOCUMENT

1. **Business requirements**
   - 1.1 Background
   - 1.2 Business opportunity
   - 1.3 Business objectives
   - 1.4 Success metrics
   - 1.5 Vision statement
   - 1.6 Business risks
   - 1.7 Business assumptions and dependencies
2. **Scope and limitations**
   - 2.1 Major features
   - 2.2 Scope of initial release
   - 2.3 Scope of subsequent releases
   - 2.4 Limitations and exclusions
3. **Business context**
   - 3.1 Stakeholder profiles
   - 3.2 Project priorities
   - 3.3 Deployment considerations

FIGURE 5-3 Suggested template for a vision and scope document.

- Keeping the scope in focus
  - Using business objectives to make scoping decisions
  - Assessing the impact of scope changes

SCOPE REPRESENTATION TECHNIQUES

- Context diagram (Context-Level Data-Flow Diagram)
  - shows the system under consideration as a single high-level process and then shows the relationship that the system has with other external entities (systems, organizational groups, external data stores, etc.).

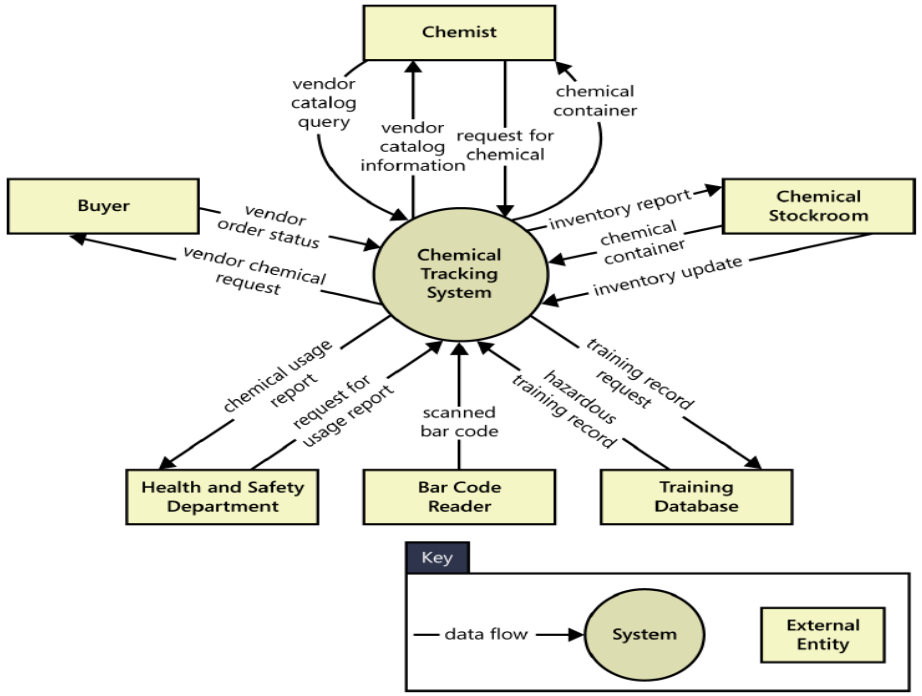

FIGURE 5-6 Partial context diagram for the Chemical Tracking System.

SCOPE REPRESENTATION TECHNIQUES

- Ecosystem map
  - shows which systems interact with each other and the nature of the relationship (ATM to Transaction DB)

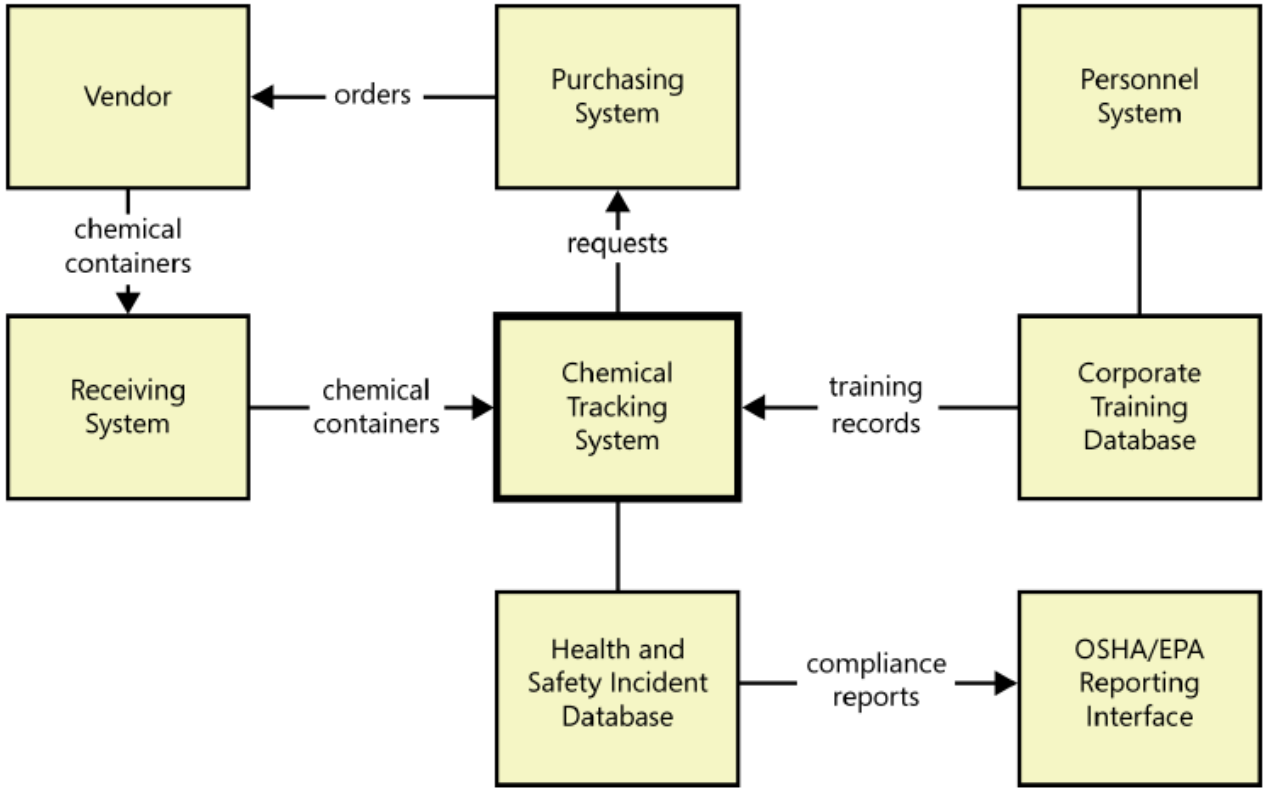

FIGURE 5-7 Partial ecosystem map for the Chemical Tracking System.

SCOPE REPRESENTATION TECHNIQUES

- Feature tree

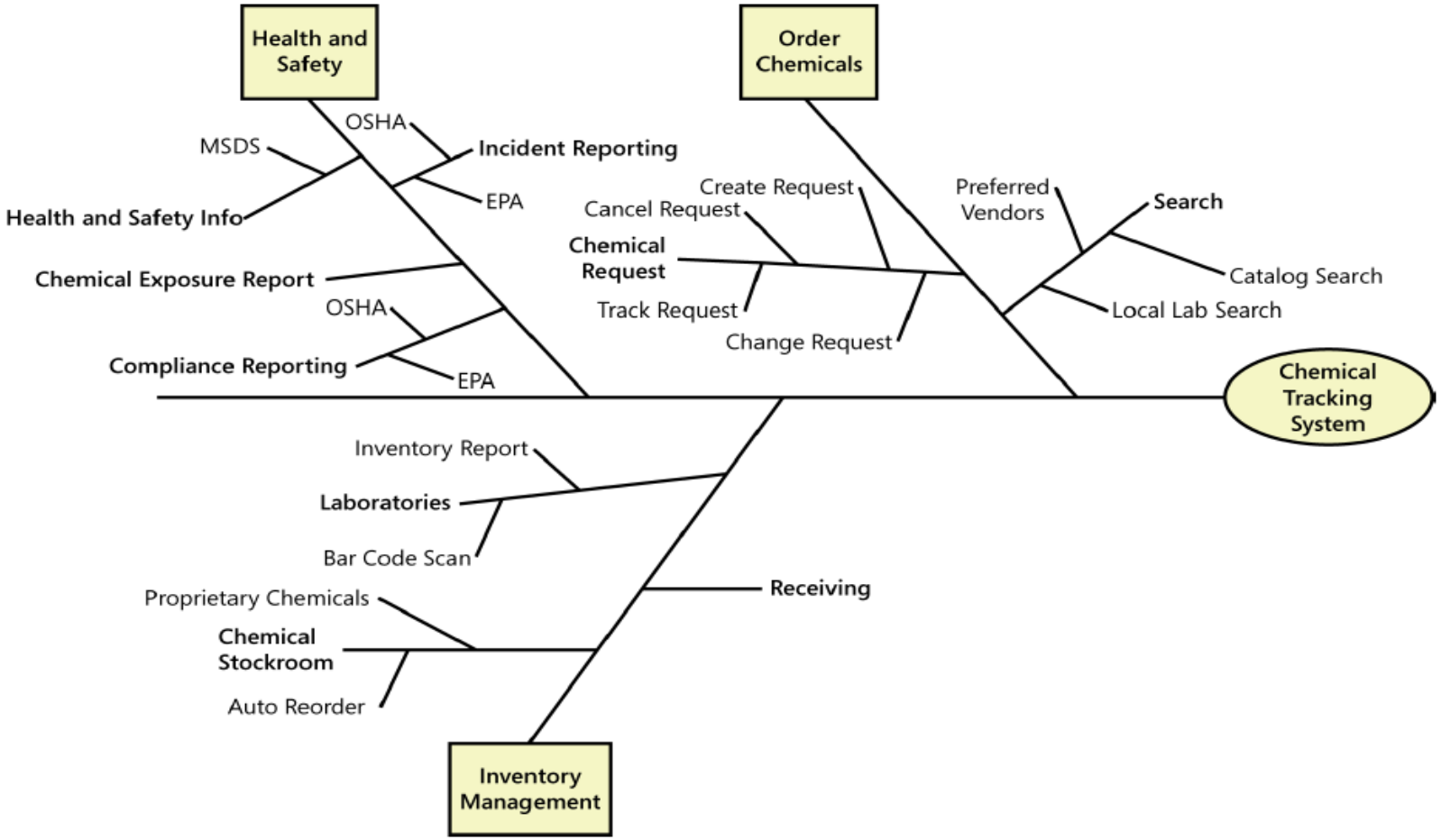

FIGURE 5-8 Partial feature tree for the Chemical Tracking System.

EVENT LIST

- An event list identifies external events that could trigger behaviour in the system.
- The event list depicts the scope boundary for the system by naming possible business events triggered by users, time-triggered (temporal) events, or signal events received from external components, such as hardware devices.
- You can check the event list against the context diagram and ecosystem map for correctness and completeness for any missing elements

External Events for Chemical Tracking System

- Chemist places a chemical request.
- Chemical container bar code is scanned.
- Time to generate OSHA compliance report arrives.
- Vendor issues new chemical catalog.
- New proprietary chemical is accessioned into system.
- Vendor indicates chemical is backordered.
- Chemist asks to generate his chemical exposure report.
- Updated material safety datasheet is received from EPA.
- New vendor is added to preferred vendor list.
- Chemical container is received from vendor.

FIGURE 5-9 Partial event list for the Chemical Tracking System.

VISION AND SCOPE ON AGILE PROJECTS

- In an agile project, development is performed in a series of fixed timebox iterations.
- The scope of each iteration consists of user stories selected from a dynamic product backlog, based on their relative priority and the estimated delivery capacity of the team for each timebox.
  - The number of iterations (the overall project duration) still depends on the total amount of functionality to be implemented, but the scope of each iteration is controlled to ensure timely completion.
- Alternatively, some agile projects fix the overall project duration, yet are willing to modify the scope.
  - The number of iterations might remain the same, but the scope addressed in remaining iterations changes according to the relative priorities of existing and newly defined user stories.

FINDING THE VOICE OF USER

- To find the voice of the user, take the following steps:
  - Identify the different classes of users for your product
  - Select and work with individuals who represent each user class and other stakeholder groups
  - Agree on who the requirements decision makers are for your project

USER CLASSES

- A product's users might differ in the following respects:
  - Their access privilege or security levels (such as ordinary user, guest user, administrator)
  - The tasks they perform during their business operations
  - The features they use
  - The frequency with which they use the product
  - Their application domain experience and computer systems expertise
  - The platforms they will be using (desktop PCs, laptop PCs, tablets, smartphones, specialized devices)
  - Their native language
  - Whether they will interact with the system directly or indirectly

CLASSIFYING USERS

Stakeholder

- Customer
  - Direct and Indirect Users
    - Favored user class: satisfaction is aligned to business goal
    - Disfavored user class: not to give access to the user (user privilege)
    - Ignored user: class product is not suit for them

IDENTIFYING YOUR USER CLASSES

- While performing stakeholder and user analysis, study the organization chart to look for:
  - Departments that participate in the business process (e.g. BP - pizza delivery system)
  - Departments that are affected by the business process
  - Departments or role names in which either direct or indirect users might be found
  - User classes that span multiple departments
  - Departments that might have an interface to external stakeholders outside the company

USER PERSONAS

- A persona is a description of a hypothetical (fictional character), generic person who serves as a stand-in for a group of users having similar characteristics and needs.
- Here's an example of a persona for one user class on the Chemical Tracking System:
  - *Fred, 41, has been a chemist at Contoso Pharmaceuticals since he received his Ph.D. 14 years ago. He doesn't have much patience with computers. Fred usually works on two projects at a time in related chemical areas. His lab contains approximately 300 bottles of chemicals and gas cylinders. On an average day, he'll need four new chemicals from the stockroom. Two of these will be commercial chemicals in stock, one will need to be ordered, and one will come from the supply of proprietary Contoso chemical samples. On occasion, Fred will need a hazardous chemical that requires special training for safe handling. When he buys a chemical for the first time, Fred wants the material safety data sheet emailed to him automatically. Each year, Fred will synthesize about 20 new proprietary chemicals to go into the stockroom. Fred wants a report of his chemical usage for the previous month to be generated automatically and sent to him by email so that he can monitor his chemical exposure.*

THE PRODUCT CHAMPION

- **Product Champions are key members of the user community who provide the requirements**
- Each product champion serves as the primary interface between members of a single user class and the project's business analyst
- The best product champions have a clear vision of the new system
- The product champion approach works best if each champion is fully empowered to make binding decisions on behalf of the user class he represents
- Known as Product owner in Agile Development

PRODUCT CHAMPION ACTIVITIES

Category	Activities
Planning	• Refine the scope and limitations of the product • Identify other systems with which to interact • Evaluate the impact of the new system on business operations • Define a transition path from current applications or manual operations • Identify relevant standards and certification requirements
Requirements	• Collect input on requirements from other users • Develop usage scenarios, use cases, and user stories • Resolve conflicts between proposed requirements within the user class • Define implementation priorities • Provide input regarding performance and other quality requirements • Work with other decision makers to resolve conflicts among requirements from different stakeholders • Provide specialized algorithms

PRODUCT CHAMPION ACTIVITIES (CNTD.)

Category	Activities
Validation and verification	• Review requirements specifications • Define acceptance criteria • Develop user acceptance tests from usage scenarios • Provide test data sets from the business • Perform beta testing or user acceptance testing (evaluate prototypes)
User aids	• Write portions of user documentation and help text • Contribute to training materials or tutorials • Demonstrate the system to peers
Change management	• Evaluate and prioritize defect corrections and enhancement requests • Dynamically adjust the scope of future releases or iterations • Evaluate the impact of proposed changes on users and business processes • Participate in making change decisions

PRODUCT CHAMPION TRAPS TO AVOID

- Managers **override the decisions** that a qualified and duly authorized product champion makes. Perhaps a manager has a wild new idea at the last minute, or thinks he knows what the users need. This behaviour often results in dissatisfied users and frustrated product champions who feel that management doesn't trust them.
- A product champion forgets that he is **representing other customers** and presents only his own requirements won't do a good job. He might be happy with the outcome, but others likely won't be.
- A product champion who **lacks a clear vision** of the new system might defer decisions to the BA. If all of the BA's ideas are fine with the champion, the champion isn't providing much help.
- A senior user might **nominate a less experienced user as champion** because she doesn't have time to do the job herself. This can lead to backseat driving from the senior user who still wishes to strongly influence the project's direction.

RESOLVING CONFLICTING REQUIREMENTS

TABLE 6-3 Suggestions for resolving requirements disputes

Disagreement between	How to resolve
Individual users	Product champion or product owner decides
User classes	Favored user class gets preference
Market segments	Segment with greatest impact on business success gets preference
Corporate customers	Business objectives dictate direction
Users and user managers	Product owner or product champion for the user class decides
Development and customers	Customers get preference, but in alignment with business objectives
Development and marketing	Marketing gets preference

REQUIREMENTS ELICITATION

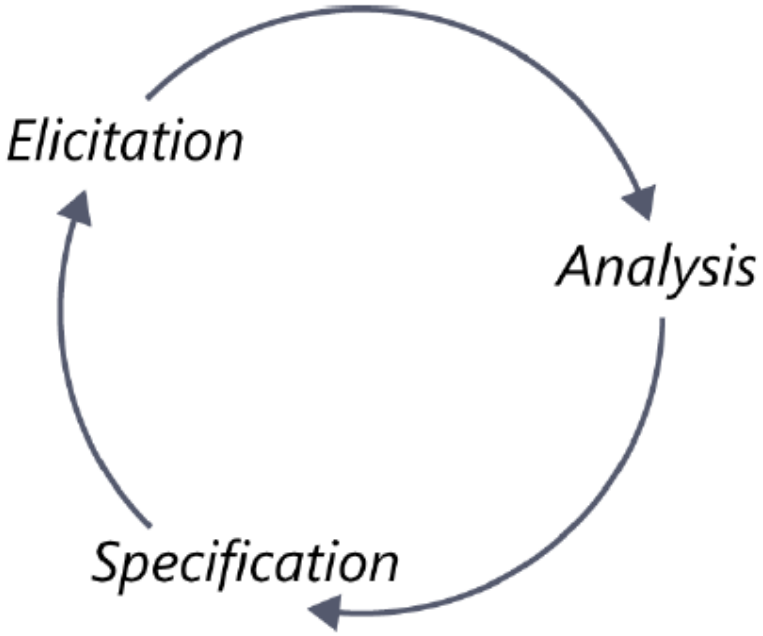

FIGURE 7-1 The cyclic nature of requirements elicitation, analysis, and specification.

ELICITATION ACTIVITIES

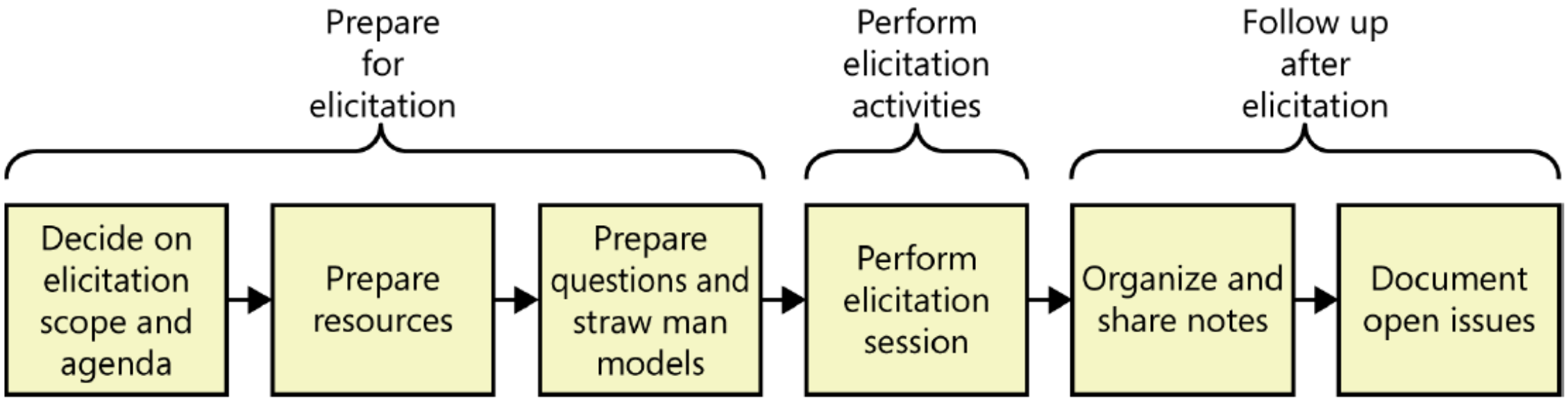

FIGURE 7-2 Activities for a single requirements elicitation session.

REQUIREMENTS ELICITATION TECHNIQUES

- **Interviews**
  - Establish rapport (understand each-other)
  - Stay in scope
  - Prepare questions
  - Suggest ideas
  - Listen actively
- **Observations**
  - Ethnography
  - System interface analysis
  - User interface analysis
  - Document analysis
- **Workshops**
  - Establish and enforce ground rules
  - Fill all of the team roles
  - Plan an agenda
  - Stay in scope
  - Use flipcharts to capture items for later consideration
  - Timebox discussions
  - Keep the team small but include the right stakeholders
  - Keep everyone engaged

REQUIREMENTS ELICITATION TECHNIQUES

- **Questionnaires**
  - Provide answer options that cover the full set of possible responses (open & closed question)
  - Make answer choices both mutually exclusive (no overlaps in numerical ranges, 1-5 &5-10)
  - Don't phrase a question in a way that implies a "correct" answer
  - If you use scales, use them consistently throughout the questionnaire (e.g. 1-9)
  - Consider consulting with an expert in questionnaire design and administration to ensure that you ask the right questions of the right people
  - Always test a questionnaire before distributing it. It's frustrating to discover too late that a question was phrased ambiguously or to realize that an important question was omitted
  - Don't ask too many questions or people won't respond

ELICITATION TECHNIQUES

	Interviews	Workshops	Focus groups	Observations	Questionnaires	System interface analysis	User interface analysis	Document analysis
Mass-market software	x		x		x			
Internal corporate software	x	x	x	x		x		x
Replacing existing system	x	x		x		x	x	x
Enhancing existing system	x	x				x	x	x
New application	x	x				x		
Packaged software implementation	x	x		x		x		x
Embedded systems	x	x				x		x
Geographically distributed stakeholders	x	x			x			

FIGURE 7-3 Suggested elicitation techniques by project characteristic.

ELICITATION PROCESS

- **Preparing for Elicitation**
  - Plan session scope and agenda
  - Prepare resources
  - Learn about the stakeholders
  - Prepare questions
  - Prepare straw man models (brain storming, business jargon)
- **Performing Elicitation Activities**
  - Educate stakeholders
  - Take good notes
  - Exploit the physical space
- **Following Up after Elicitation**
  - Organizing and sharing the notes (check with the customer again)
  - Documenting open issues

CLASSIFYING CUSTOMER INPUT

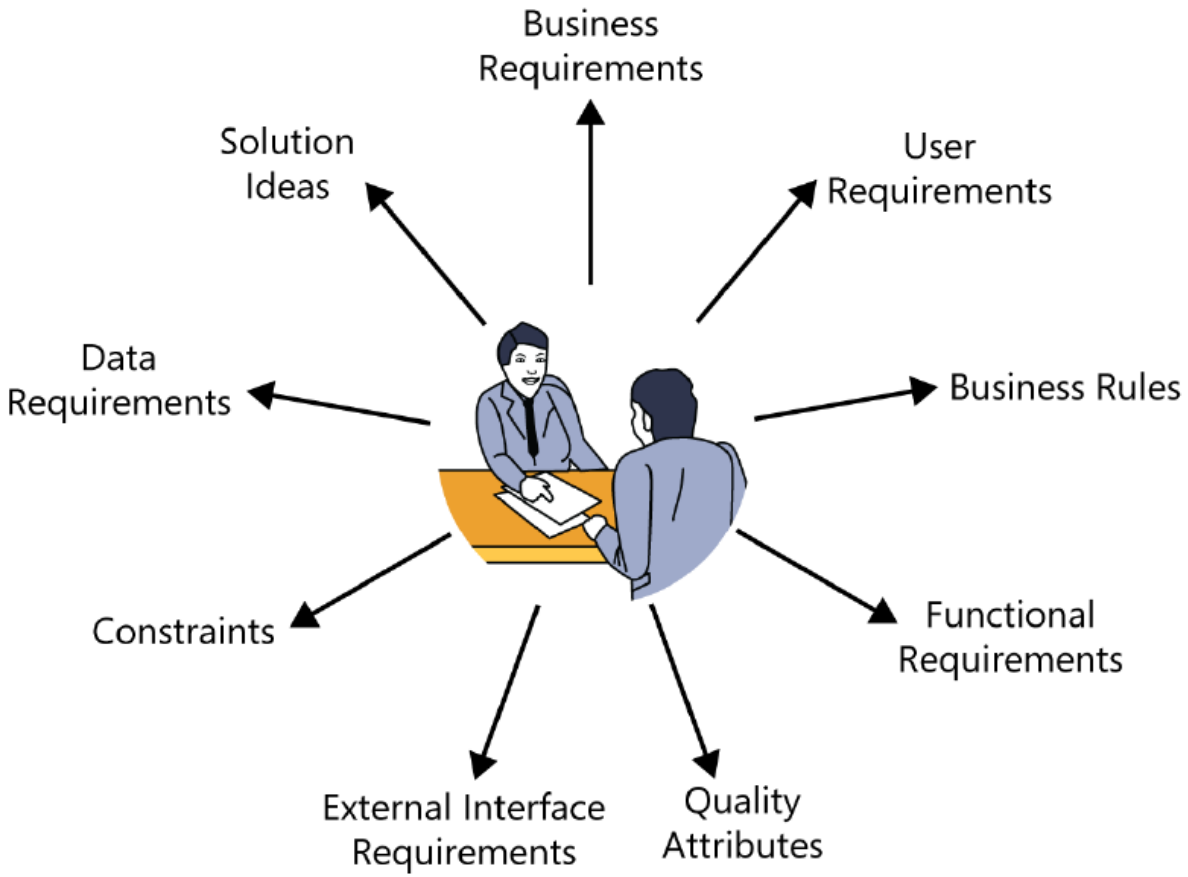

FIGURE 7-7 Classifying customer input.

CLASSIFYING CUSTOMER INPUT

- Anything that doesn't fit into the classified categories might be:
  - A project requirement not related to the software product, such as the need to train users on the new system
  - A project constraint, such as a cost or schedule restriction (as opposed to the design or implementation constraints)
  - Additional information of a historical, context-setting, or descriptive nature
  - Extraneous information that does not add value

HOW DO YOU KNOW WHEN YOU'RE DONE?

- The users can't think of any more use cases or user stories. Users tend to identify user requirements in sequence of decreasing importance.
- Users propose new scenarios, but they don't lead to any new functional requirements. A "new" use case might really be an alternative flow for a use case you've already captured.
- Users repeat issues they already covered in previous discussions.
- Suggested new features, user requirements, or functional requirements are all deemed to be out of scope.
- The users are proposing capabilities that might be included "sometime in the lifetime of the product" rather than "in the specific product we're talking about right now."

SOME CAUTIONS ABOUT ELICITATION

- Balance stakeholder
  - Collect input from too few representatives
  - Hearing the voice of the loudest (most opinionated customer)
- Define scope appropriately
  - Project scope is improperly defined
  - Requirements should specify 'what' the system do (requirements) but not 'how' to do (process)
- Research within reason
  - Exploratory research sometimes disrupts elicitation (but extensive research)

ASSUMED AND IMPLIED REQUIREMENTS

- Assumed requirements
  - are those that people expect without having explicitly expressed them
  - what you assume as being obvious might not be the same as assumptions that various developers make

- Implied requirements
  - are necessary because of another requirement but aren't explicitly stated (dependency)
  - Developers can't implement functionality they don't know about

FINDING MISSING REQUIREMENTS

- Decompose high-level requirements into enough detail to reveal exactly what is being requested
- Ensure that all user classes have provided input
- Trace system requirements, user requirements, and business rules to their corresponding functional requirements to make sure that all the necessary functionality was derived
- Sets of requirements with complex Boolean logic (ANDs, ORs, and NOTs) often are incomplete
- Create a checklist of common functional areas to consider for your projects
- A data model can reveal missing functionality (ER diagram)

USE CASES & USER STORIES

- A use case
  - Describes a sequence of interactions between a system and an external actor that results in the actor being able to achieve some outcome of value.
  - The names of use cases are always written in the form of a verb followed by an object.

- User stories
  - Often are written according to the following template, although other styles also are used:
  - *As a <type of user>, I want <some goal> so that <some reason>*

USE CASES & USER STORIES

Sample Use Cases

Application	Sample use case
Online bookstore	Update Customer Profile Search for an Item Buy an Item Track a Shipped Package Cancel an Unshipped Order

Sample User Story

Application	Sample use case	Corresponding user story
Online bookstore	Update Customer Profile	As a customer, I want to update my customer profile so that future purchases are billed to a new credit card number.

THE USE CASE APPROACH

- **The essential elements of a use case are the following:**
  - A unique identifier and a concise name that states the user goal
  - A brief textual description that describes the purpose of the use case
  - A trigger condition that initiates execution of the use case
  - Zero or more preconditions that must be satisfied before the use case can begin
  - One or more post-conditions that describe the state of the system after the use case is successfully completed
  - A numbered list of steps that shows the sequence of interactions between the actor and the system—a dialog—that leads from the preconditions to the post-conditions

IDENTIFY ACTORS OF USE-CASE

- Following are some questions you might ask to help user representatives identify actors:
  - Who (or what) is notified when something occurs within the system?
  - Who (or what) provides information or services to the system?
  - Who (or what) helps the system respond to and complete a task?

IDENTIFYING USE CASES

- Identify the actors first, then lay out the business processes being supported by the system, and define the use cases for activities where actors and systems interact.
- Create a specific scenario to illustrate each business process, then generalize the scenarios into use cases and identify the actors involved in each one.
- Using a business process description, ask, "What tasks must the system perform to complete this process or convert the inputs into outputs?" Those tasks might be use cases.
- Identify the external events to which the system must respond, then relate these events to participating actors and specific use cases.
- Use a **CRUD analysis** to identify data entities that require use cases to **C**reate, **R**ead, **U**pdate, **D**elete, or otherwise manipulate them.
- Examine the context diagram and ask, "What objectives do each of these external entities want to achieve with the help of the system?"

USE CASE TRAPS TO AVOID

- Too many use cases
- Highly complex use cases
- Including data definitions in the use cases
- Use cases that users don't understand

REFERENCES

- Wiegers, K., & Beatty, J. (2013). *Software requirements*. Pearson Education