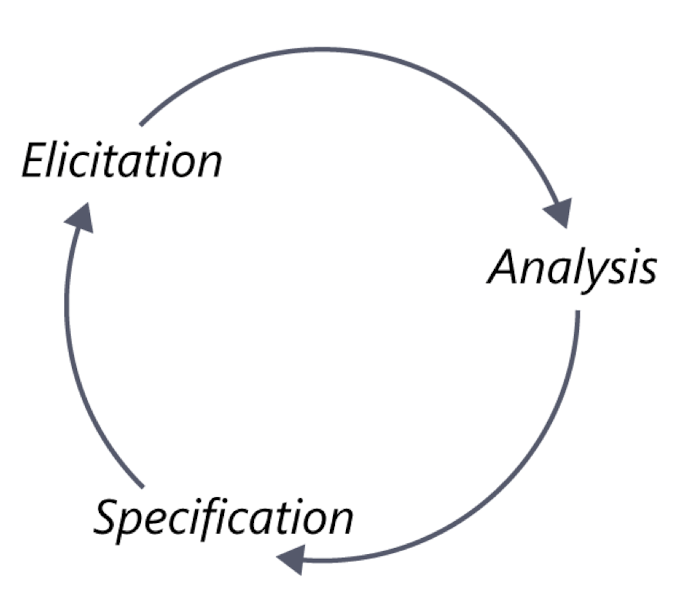
**Chapter 7: Requirements Elicitation**

Central idea: identifying needs before starting building the system

Understanding the thought processes behind the requirements in users state, extract user’s underlying logic. In this process, you could use the vocabulary of the business domain instead of forcing customers to understand technical terms. The output of requirement development is a common understanding of the needs held by the diverse project stakeholders. 

The nature of requirements development is cyclic

During this process, you have to interact with stakeholders to elicit requirements, and you also have to work on your own discovery information.

Most projects will user a combination of both facilitated and independent elicitation activities.

Interview: the most straightforward way is to ask users.

1. Establish rapport: introduce yourself if the attendees do not know you. Then, review the agenda, remind attendees of the session objectives, and address preliminary questions
2. Stay in scope: keep the discussion focused on its scope
3. Prepare questions and strawman models: prepare a list of questions to guide the conversation, and also prepare straw man models.
4. Suggest ideas:
5. Listen actively:

Workshops: facilitated sessions with multiple stakeholders and formal roles, encouraging stakeholders collaboration in defining requirements. It could be resource intensive, materials should be prepared to avoid time wasting.

1. Establish and enforce ground rules: reach an agreement on some basic operating principles.
2. Fill all of the team roles: note taking, time keeping, scope management, ground rule management are required.
3. Plan an agenda: a clear plan, which should be prepared ahead of the workshop
4. Stay in scope: keep each workshop focused on the right level of abstraction for that session’s objective. Be careful for those detailed discussions that are off-topic.
5. Use parking lots to capture items for later consideration: an array of random but important informations, like quality attributes, business rules, user interface, ideas, and more. Using flipcards to organize this info. Just record these until they stop the process.
6. Timebox discussion: allocating a fixed period of time for each topic.
7. Keep team small but include the right stakeholders: small group work faster. 5-6 members, and run multiple groups parallel for different user classes.
8. Keep team engaged: ensure everyone is heard and everyone is giving idea.

Focus groups: focus group sessions must be interactive, allowing all users a chance to voice thor thoughts. When conflict erupt, first looking for nonverbal clues. If someone will not participate in a productive way, determine whether he would prevent the process. If so, thank him for his time. Focus groups must be facilitated. Keep them on topic, do not influencing the opinion. Recording for feedbacks.

Observation: time consuming, may not suitable for all tasks. Keep the observation time within 2 hours, and pick important or high-risk task and multiple user classes for observation. It could help validating information collected from other sources, to identify new topics for interviews, and get a new workflow. Observation could be silent or interactive, silent is good for busy users cannot be interrupted, while interactive could interrupt the user mid-task and ask a question.

Questionnaires: used to survey a large group of users. Preparing a well-written questions is the biggest challenge.

1. Provide answer options that cover the full set of possible answers.
2. Make answer choice both mutually exclusive, no overlap in numerical ranges, and exhaustive
3. Do not phrase a question in a war that implies a “correct” answer
4. Keep scales consistently throughout the questionnaire.
5. For statistical records, use closed questions with two or more specific choices.
6. Consider consulting with an expert in questionnaire design
7. Test questionnaire before distributing it.
8. Do not ask too many questions.

System interface analysis: identify functionality in the other system that would use your system.

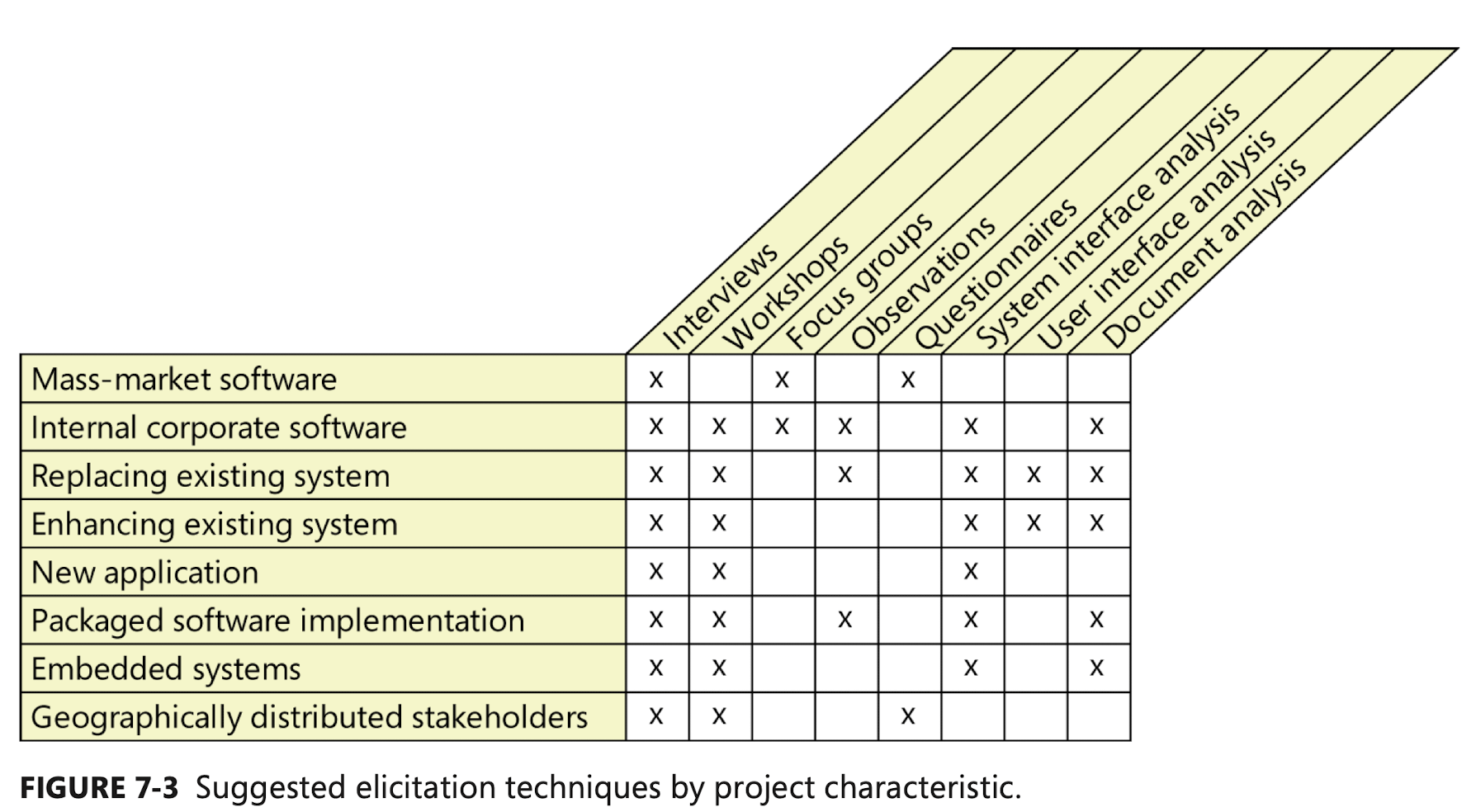
User interface analysis: interact with the existing systems directly if possible. Do not assume that certain functionality is needed in the new system just because you found it in an existing one.

Document analysis: examining any existing documentation for potential software requirements. Documents can describe corporate or industry standards that must be followed or regulations that must be complied. Past documentation could be used to check functionalities. And it could be used to describe the needed functionality. One potential risk is that the documentation is not up to date.

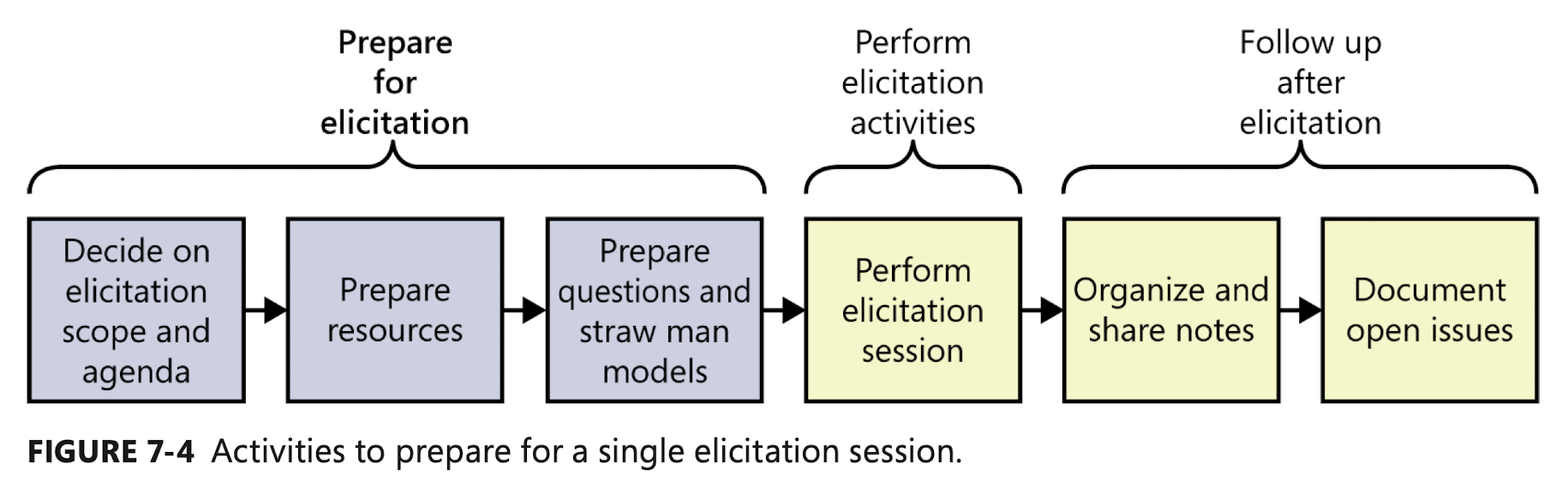
Planning elicitation on project:

The business analyst should plan the project’s approach to requirements elicitation in the early stage of the project. An elicitation plan should include the used technique, when to use them, and for what purpose. Use it as a guide and you could change it anytime.

1. Elicitation objective: plan the objective for the entire project and objectives for each planned elicitation activity.
2. Elicitation strategy and planned techniques: decide which techniques to use with different stakeholder groups. Use a combination of the techniques above.
3. Schedule and resource estimates:
4. Documents and systems needed for independent elicitation: identify material you needed
5. Expected products of elicitation efforts
6. Elicitation risks: identify factors that could impede your ability to complete the elicitation activities, and estimate the severity of each risk, and decide how you can mitigate or control it



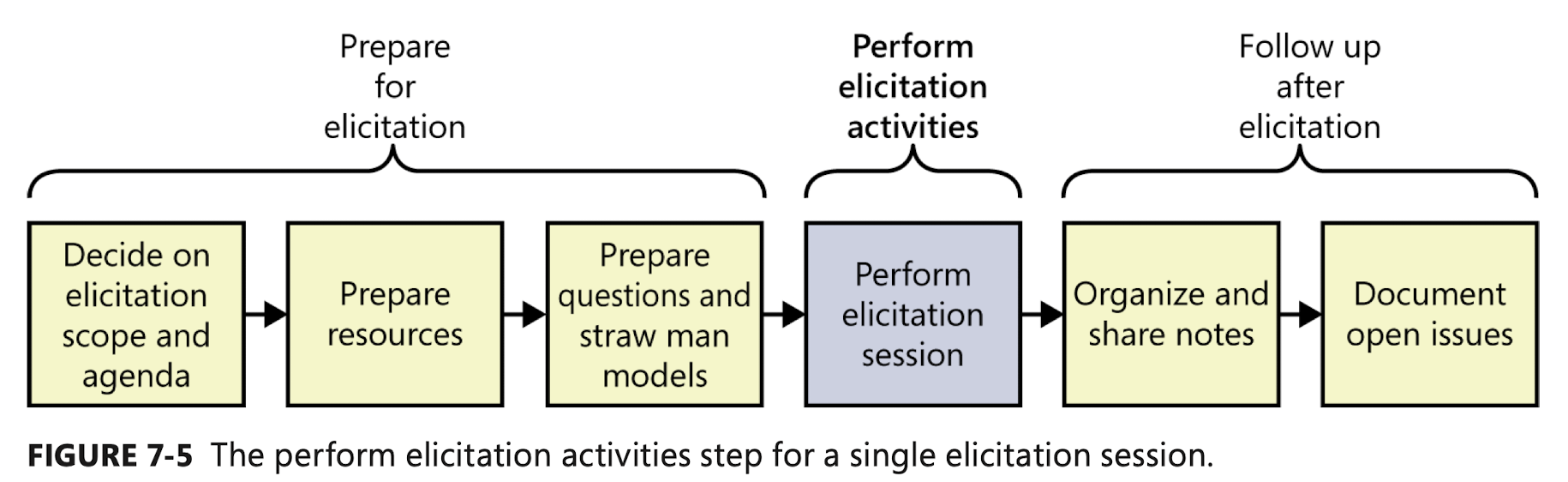
**Preparing for elicitation: make the best use of everyone’s time.**



Giving scope of the session, communicating an agenda, preparing questions, and draft materials.

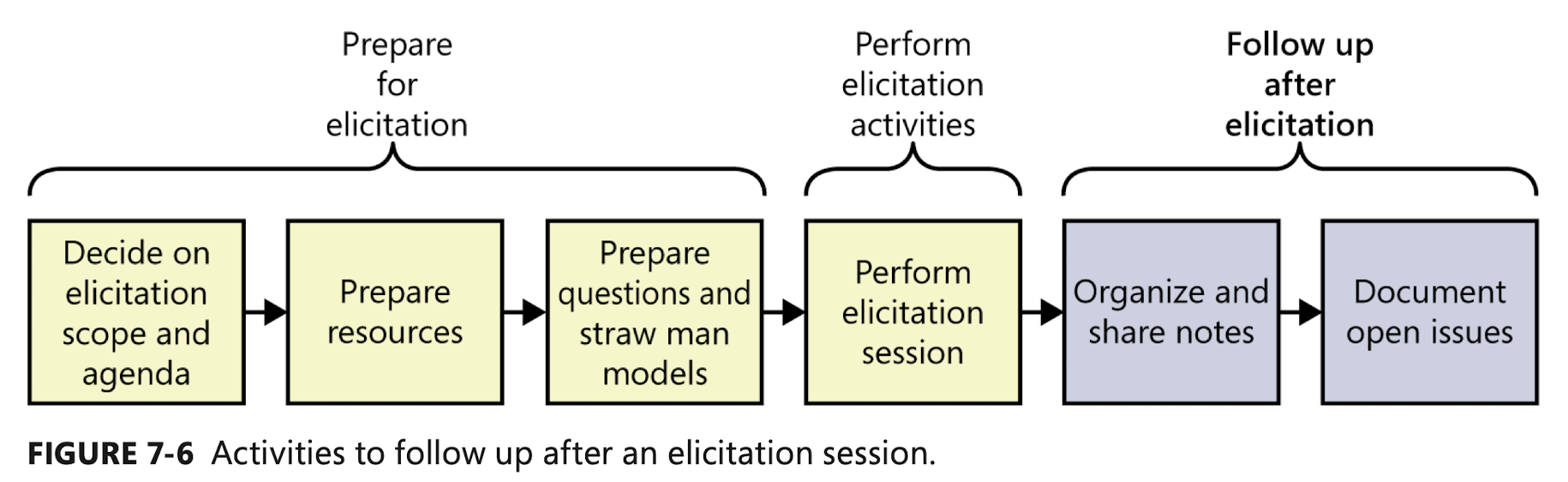
1. Plan session scope and agenda: taking into account how valuable the time is. The agenda should itemize what topics will be covered, and the available time for each topic, and target objectives. Remember to share the agenda with stakeholders in advance.
2. Prepare resources: schedule the physical resources needed, including rooms or equipment. Schedule the participants, collect documentations from various sources.
3. Learn about the stakeholders: avoid creating an “us” versus “them” tension.
4. Prepare questions: get facilitated elicitation sessions with a set of prepared questions, phrase the questions to avoid leading customers down an unintended path or toward a specific answer. Ask open-ended questions to understand the user’s current business. Probe around the exception.

Prepare straw man models: analysis models can be used during elicitation sessions like use cases and process flows. A straw man serves as a starting point for learning about a topic.



Performing elicitation activities:

1. Educate stakeholders: teach them your elicitation approach and why you choose it. And inform them that you will capture their information and send them materials for review after the session.
2. Take notes: assign someone to be the scribe, or use a record device.
3. Exploit the physical space: use walls to draw diagrams or create lists, and ask other participants to contribute to the wall.

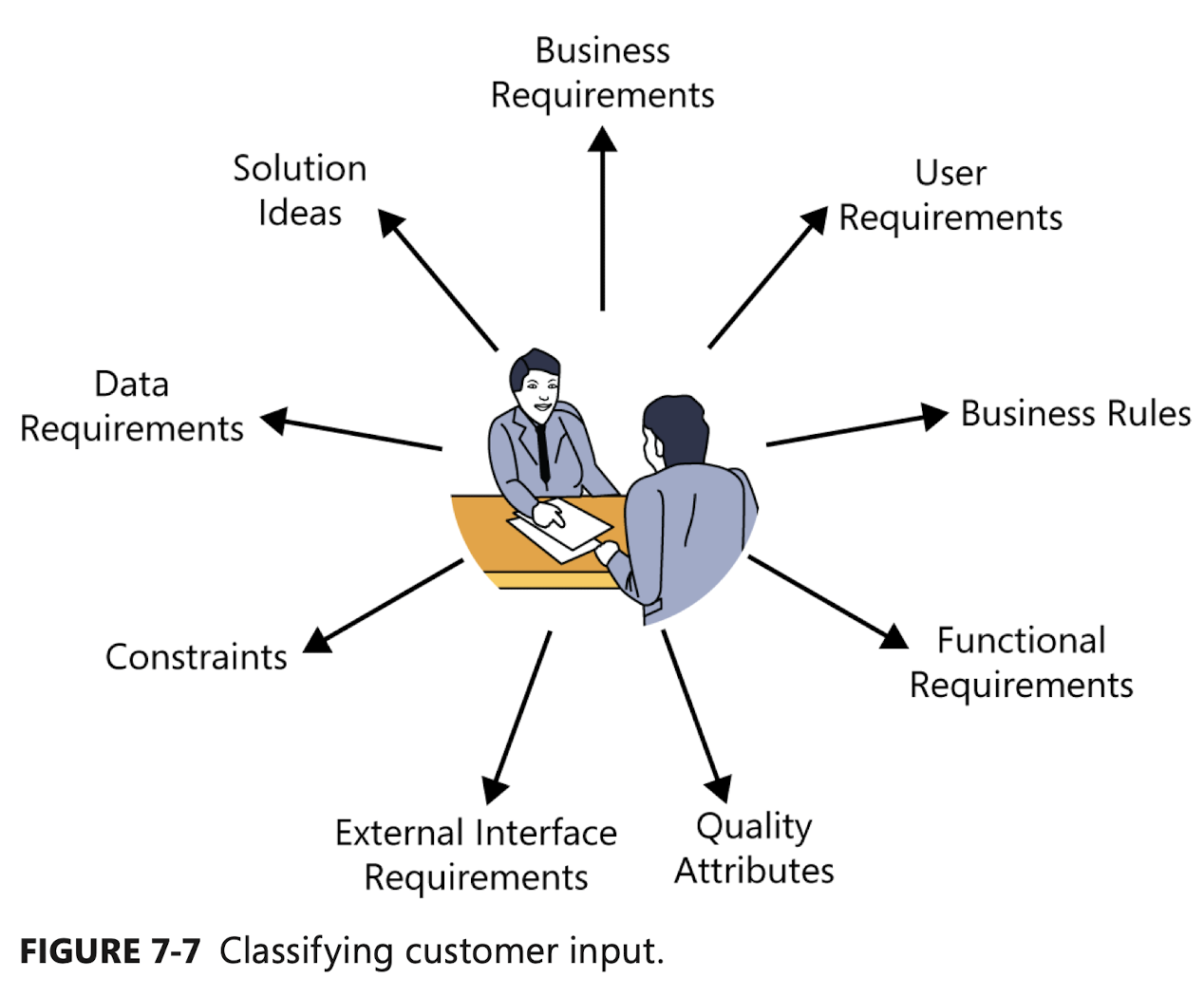


Following up after elicitation: organize and share notes, document open issues, and classify the newly gathered information.

Organizing and sharing the notes: review and update the note soon after the session is complete. Editing the elicitation notes is a risk, so keeping a set of the raw notes to refer later. Share the consolidated notes with the participants and ask them to review them to ensure that they accurately represent the session. Consider sharing the consolidated notes with other project stakeholders who weren’t present.

Documenting open issues:

Examine any parking lots from elicitation sessions for issues that are still open and record them in an issue-tracking pool. Also record relevant notes, progress made, and owner, and a due date.



Classifying customer input: if there is something does not fit into this graph, it could be:

1. Project requirement not related to the software development
2. Project constraint
3. Assumption or a dependency
4. Additional information of a historical, context-setting or descriptive nature
5. Extraneous information that does not add value

You have to determine that type of information each statement you get.

1. Business requirements: financial, marketplace, business benefit
2. User requirement: “As a xxx, I need to do something”
3. Business rules: some customer says that only certain users can perform an activity under specific conditions, like “must comply with”, “if … then …”
4. Functional requirements: describe the observable behaviors the system will exhibit under some condition and some action system would ask user to take, like “if….then….”
5. Quality attributes: describe how well the system does something, like “open the page quick”
6. External interface requirements: describe the connections with other things, like “sending message to …”, “receiving from …”
7. Constrains: device physical properties, interface connection, like “must written in cpp”, or “file could not be larger than 1GB”
8. Data requirements: specifying the format, type, values of data storage
9. Solution ideas: some “requirements” are actually suggested solutions. Keep asking user “why” to get the true need.

How do you know when you’re done?

1. The user cannot think of any more use cases or user stories. Users tend to identify user requirements in sequence of decreasing importance
2. User propose new scenarios, but they do not lead to any new functional requirements. A “new” use case might really be an alternative flow for a use case you have already captured
3. User repeat issues they already covered in previous discussion