Ambiguity and Architectural Questions

Group Assignment

Homework Assignment:

You are the architect. For the following general system description, write down questions you will ask the analysts. These are to clarify the “requirements” and add to them. Write down as many as you can think of.

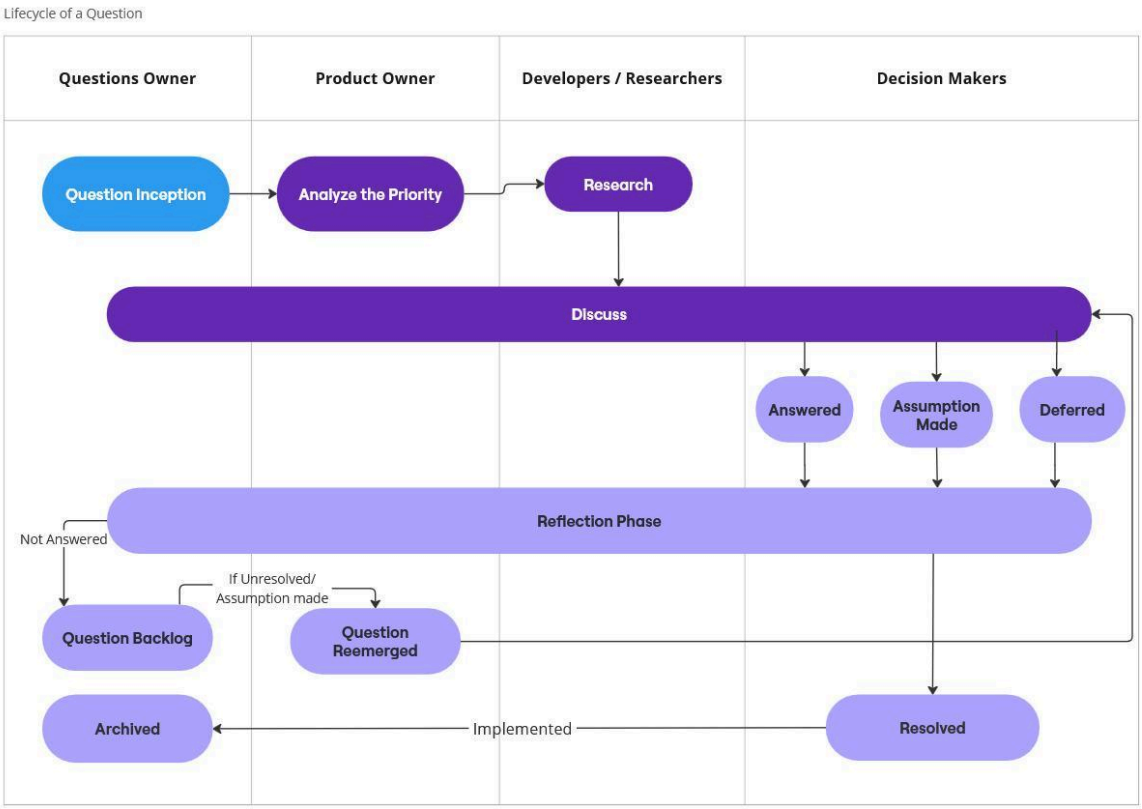
You should have at least 20 questions.

System description

In software architecture, natural questions are the spontaneous, essential questions that emerge throughout the development lifecycle, shaping key decisions and influencing project outcomes. Despite their importance, these questions are often mismanaged, leading to knowledge loss, duplication of effort, drift, inefficient resource use, and difficulties for outsiders trying to understand project specifics.

Given these challenges, we need a set of tools seamlessly integrated (a toolset) to support and manage the questions lifecycle and related knowledge that arise during software architecture design. It should be good to support one single person, one team, or many teams working collaboratively.

We identified key activities within the lifecycle and now we want to design tools (AI or not) to enhance collaboration and preserve architectural knowledge more effectively, outperforming traditional methods in efficiency and precision. Those tools will support questions storage, management, sharing, querying, knowledge recovery, well, you got the idea.



Part 1:

[30min] Work with your team to think about the system and write down your questions. Each question should be expressed as a full sentence. Make sure your writing is clear and easy to understand. Although the focus is on requirements, you will find yourself drifting into the solution space at times. That's fine. Remember, the spaces interconnect!

[30min] Merge and organize your questions with those of the other teams of your class [30min] along two dimensions:

- Some logical grouping of topics, in some way that makes sense to you. It might be by features, quality attributes, logical components, or even combinations of them.
- Time when the questions need to be answered.

The easiest way to organize them is to get a bunch of yellow sticky notes and write a question on a sticky note. Then organize them on a spiderweb drawn on a white board. The radii going out are the logical categories, and distance from the center is time (the middle is "now".) You may have some questions that fall between two categories; just place them where it makes sense. Then take a picture of the *spider web* and include it in a document..

[30min] Considering the five topmost important questions, design the architecture of the overall system. Prepare a **5 minute** presentation of your questions and architecture. Send the document and presentation to your teachers.

[30min] Present your architecture, so far.

Part 2:

After the class, refine your architecture trying to address more questions and making it as simple and elegant as possible. Intuitiveness, usability, user experience, easy to learn, quick, powerful, retrievability, always at hand, are some of the quality attributes we look for in the system.

Hints and Considerations

- Don’t go into details about the data. Just start with the general types.
- What are the different ways people can use the toolset system? What are the architectural implications?
- What are the major modules/tools? Which connect with which others? Considerations:
    - Separation of Concerns (similar to responsibilities for each module)
    - Appropriate coupling
- What are typical scenarios? Walk through a couple.

Document your Design

- Box and line charts
    - Be sure that you have enough detail. Two or three boxes is certainly not enough! You want the boxes and lines to guide the implementers.
    - Think about the layout. It should make sense. If you have different sizes or styles of boxes, why?
    - Make sure the meaning of the lines are clear. We will talk about connectors later; right now do the best you can. Consider labels on the lines.
- Text explanations
    - Explain all the diagrams.
    - It is often good to explain the rationale behind decisions you made. For example, there are some intentional ambiguities in the spec. (Just like real life.) You might want to explain why you resolve ambiguities the way you did.
    - The goal is that the diagrams and text provide enough information that a team could implement the system.