QuickTip – How to Gather Requirements

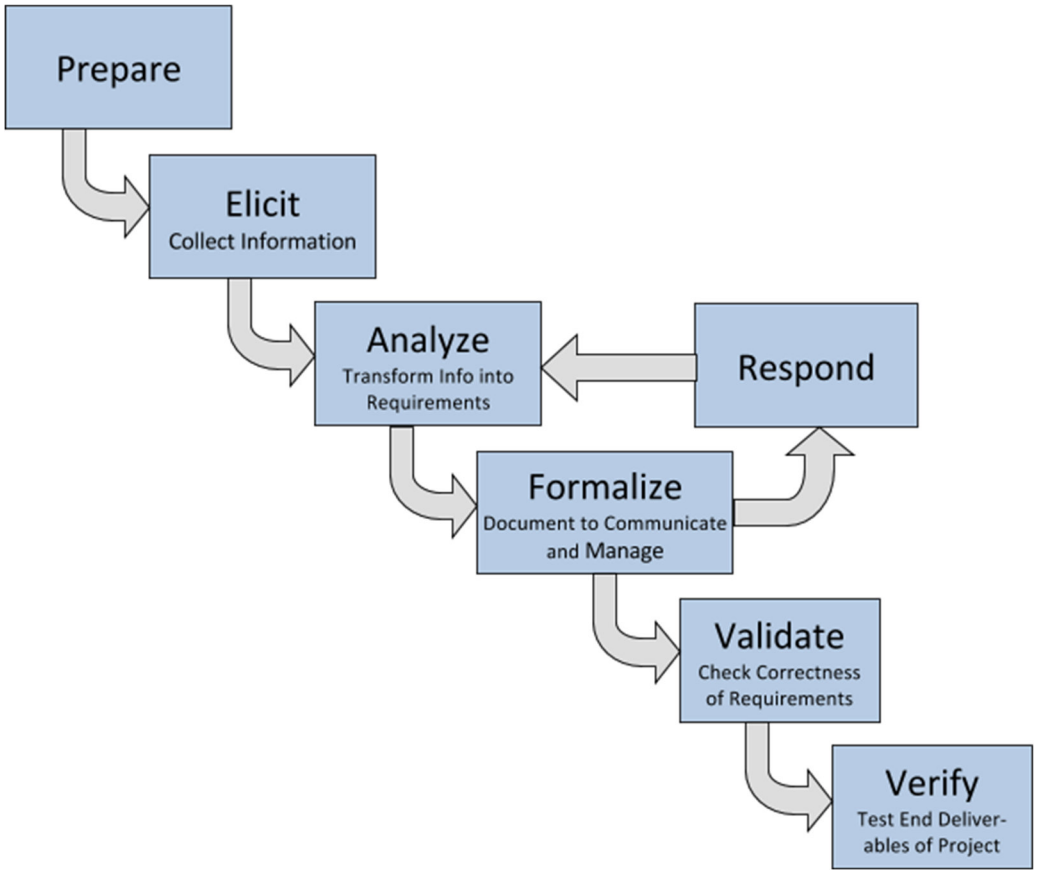
Create good requirements that define what your project will deliver.

When to Use

For many projects, the end deliverables can be sufficiently defined in the project charter and plan. However, some large or complex projects have broad deliverables. On those projects it is helpful to refine the deliverables into a set of detailed requirements during planning. Requirements give the project team the details they need to be able to create an accurate breakdown and ordering of the work and project plan. Use this tip when you need to create and manage a set of requirements.

Procedure

The diagram below shows the typical steps in the process of managing requirements. The first six steps happen during the Plan phase of a project. The final step (Verify) is done during the Execute or Close phase. Adapt these steps to match the size and complexity of your project.



1. Prepare: Identify where requirements will come from. Prepare to talk with people in those areas by learning about the vocabulary and workflows they use to do their work.

2. Elicit: Collect relevant information from as many users and stakeholders as is feasible. Many people use the acronym LCP as a reminder of how to interview a stakeholder. First listen carefully, then confirm what you heard, and finally probe to get a deeper understanding.

Consider providing mockups or straw proposals that users can react to. This may help them clearly state what they need. Observation can help, too, since the best understanding of what a user needs often comes by observing them doing their job.

3. Analyze: Study the information you've collected in order to discern true requirements. Modeling techniques can help out in this step. Examples are process flow diagrams, gap (as-is and to-be) analysis, system and data flow diagrams to show interfaces, information flow and storage between people and systems, and KJ (conjoint) analysis or affinity diagramming to find commonalities.

4. Formalize: Record the emerging requirements so the stakeholders can review, modify, and approve them. This includes prioritizing the requirements so that the project team can make smart tradeoffs during the project. A typical approach is to use three priority categories: required, highly desired, and "would be nice." Allow only a minority of requirements in the required category.

5. Respond: Provide feedback based on the requirements. Tell stakeholders what your project can actually accomplish. Help them understand tradeoffs and work through options so they can make important tradeoffs early in the program, when flexibility is highest. If this changes the requirements, it may be necessary to iterate through the Analysis and Formalize steps again.

6. Validate: Confirm that the requirements are correct and that the stakeholders agree that the requirements are good enough. Then, use the validated requirements to plan and execute the project. Update them as the project progresses, making sure to communicate changes to affected people.

7. Verify: When the project's deliverables are ready, verify that they meet the requirements. This is often done during testing.

On very large projects, requirements engineering is done iteratively, capturing and getting agreement on high level requirements first, then refining them into more detailed requirements.

Considerations

1. A requirement describes a feature, function, or deliverable. It describes *what*, not *how*.
2. Requirements should have these characteristics:
    a. Complete and consistent
    b. Unambiguous
    c. Specific
    d. Verifiable
    e. Feasible
    f. Prioritized
3. Facilitate discussions among members of the project team to get an initial list of requirements.
4. Balance the needs and desires of the users with the constraints and judgment of the project team.
5. Be prepared to iterate through the list of requirements several times to refine it and get agreement.
6. Try to cover the breadth of important aspects of what you will be creating, including reliability, standards compliance, performance, usability, and maintainability.

Example

Here is an example for a project that is supposed to make improvements to a registration system that is used for conferences and classes.

Well-written:

"7.1 An attendee shall receive written confirmation of their registration details from the registration system within one hour of successful pre-registration. (Priority = highly desired)"

Poorly written:

"Send email during registration."

Why is this poorly written?

1. Requiring email is too focused on the how-to. For example, a fax might be a better solution.
2. It is ambiguous and incomplete. Who should the email be sent to? When? What should its content be? Who or what sends it? Because of the ambiguity, it will also be hard to verify.

3. It is not prioritized, so it is not possible to determine how important it is relative to other requirements. This will make tradeoffs difficult if not all requirements can be fully implemented.

Poor requirements gathering can break a project. If you are leading a complex project, consider getting a requirements expert on your team or requesting coaching or external review of the requirements.