UNIVERSITY OF MARYLAND
University College

Learning Resource

Developing Requirements for an IT System

Where Do the Requirements Come From?

Let's assume that someone in the organization identifies one or more problems with the way a process is working. Whether the current process is supported by an IT system or not, the analyst might ask people with different roles in the process two questions:

- What problems are you having in performing the task today?
- How do you see an IT system helping to improve things?

These questions should elicit a variety of responses from multiple perspectives. The executives might answer with how the organizational strategies and objectives could be better supported with an IT system. Managers may answer the questions with how an IT system would help them manage the people and processes better. Front-line employees will likely focus on their tasks and which steps could be done more easily and quickly if they had a system. The analyst will use information gathered during the process analysis phase to help stakeholders identify and clarify what the system needs to do for them.

If there is organizational agreement that a new system is probably needed, then a determination should be made as to whether a system will need to be developed or if a pre-built commercial off-the-shelf (COTS) solution might work. This would include answering the following types of questions:

- For what major functions or tasks is the user seeking an IT solution?
- Is there any part of that task that is likely to be unique to this organization?
- Would it be possible to find a COTS solution, since those are already created, are ready to be used, and are often much less costly to implement?

If the organization does not employ any significantly unique functions to accomplish a standard business process, then it is likely that a COTS solution exists that could meet the needs. The determination of whether a system is to be built or bought drives the level of detail needed in the requirements. Many more requirements with much more detail are needed for building a system than for buying one.

Regardless of whether a system is to be built or bought, the next step is to identify the **high level user requirements** (or “functional” requirements). This is done by interviewing the expected users of the system. Users very often know some of what they need the system to do, but are unable to list all the functions they need. One way the analyst elicits the requirements is by asking a variety of users at different levels of the organization and with different responsibilities how the processes are currently being done and what it is that the current system or process does or does not do efficiently. The manager's perspective and needs are quite different from the front-line employee trying to perform specific tasks, and the executive's perspectives and needs are unique to that level of the organization. After a series of interviews, the analyst can categorize and document the requirements that are emerging. Some of these will likely be at a very high level (e.g., "I need annual financial reports") to very low-level detailed items (e.g., "the zip code must include all 9 digits"). For an accounting system, the high-level requirements might include "the system must implement the Generally Accepted Accounting Principles (GAAP)" or "the system must produce a monthly expense statement," along with many other functions identified by the users. One of the biggest challenges for the analyst is to differentiate between a "must have" (**essential**) requirement and a "nice to have" feature. When requirements are collected and documented they are often put into these two categories. The analyst asks the end user to determine whether each requirement is a "must have" or a "nice to have" item, and documents accordingly.

Some users may identify requirements that they believe the system must perform, but that the analyst does not believe should be part of the specification for the system in question. At this point in the process, all of the requirements identified by any of the participants should be listed. Eventually, the full list of requirements will be reviewed, modified as necessary and approved by the system "owner" and major stakeholders. During that part of the process, final determinations will be made about which requirements are essential, which are "nice to have," and which should be eliminated. The list of essential requirements will be used to identify whether there are COTS products available that should be considered; "nice to have" requirements will be used to compare solutions that meet the essential requirements. In a system development environment, the essential requirements will be used to determine the scope of the project. It is often easier and less costly to include "nice to have" items in systems being developed in-house, but the overall cost of developing and maintaining IT systems must be considered in making that decision. In the systems development life cycle (SDLC) analysis phase, the project sponsor signs off on the requirements document. In later SDLC phases, the requirements are used to design, develop, and test the system.

A separate set of **system performance (system quality and security) requirements** comes from the combination of end user needs as well as technical specifications developed by the IT department. The answers, again, are elicited via interviews with expected system users and managers. Below are example questions that the analyst might ask to develop system performance requirements in each of the system quality and security categories:

- **Usability**—Do you want the system user to have access to an online help manual? Do you want the user to be able to access context-specific help while entering each data field on the screen?
- **Scalability**—How many users and how many records/transactions do you need the system to be able to accommodate? How much might those increase over time?
- **Availability**—Are there any time blocks where access to the system is not needed (e.g., no one would use the system between midnight to 4 a.m. daily)?
- **Reliability**—Can you provide examples of tasks where the system needs to create and maintain accurate/correct data?
- **Maintainability**—Are system security updates applied within 24 hours? (While end users are affected by the maintainability of the system, it is usually up to the IT department to determine whether the process used accommodates changes as needed and whether updates are made in a timely manner.)
- **Portability**—What devices do you want the users of the system to be able to use? Is it likely that they would use a smartphone, tablet, etc., to either query or use the system?
- **Interoperability**—Are there any systems with which the new system will need to directly exchange data?
- **Security**—This is another area where users are affected, but need assistance from technical specialists to determine the requirements. The analyst might ask: How sensitive is the data? Are there any regulations concerning protecting the type of data in this system (personally identifiable information, health care or other data protected by law, etc.)? Do you want users to be restricted as to what they can do with the system or what data they can access? Should this be based on their role in the organization? How often does the data change? How long could you continue to operate if the system were unavailable?

The User's Role—Identifying Requirements

As discussed above, it is the responsibility of the system users to identify the need for a solution to a problem or to identify processes that could be improved and performed more effectively or efficiently. The user is familiar with the business process to be accomplished and with how it is currently performed, and can identify any issues that exist. Previous work completed on process analysis is an important precursor to defining requirements. It is not unusual for the business person to look around and find potential IT solutions to their problems, and some want to jump immediately into acquiring a specific solution. However, without a set of requirements that has been approved by the organization, a solution that fits one set of problems may not fit the needs of other users of the system.

The Analyst's Role—Documenting Requirements

One of the business analyst's biggest challenges is to get the users to identify their requirements rather than focus on a specific solution. The analyst conducts interviews and observes the process as it exists and documents the process. Using the process analysis work done previously and by asking the types of questions discussed above, the analyst gathers the requirements for the new or updated IT system and begins to document them.

How Are Requirements Statements Written?

There are a number of "rules" for writing requirements statements. These rules help to ensure that the requirements can be clearly understood and that it is possible to determine whether or not the new system meets each of the requirements. Poorly written requirements lead to misunderstanding and misinterpretation and can lead to a system that does not do what the users need it to do.

The analyst uses the list of requirements that the users identified and rewrites each requirement to meet the criteria listed below.

Each requirement statement:

- Either describes a **task** that the user needs the system to perform, **or** states a **system performance** expectation.
- Identifies **only one** requirement; avoids the words "and," "also," "with," and "or."
- Is a **complete sentence**, with a subject (usually "the system") and predicate (intended result, action or condition).
- Uses "**must**" (not "may" or "should" or "will" or "shall"); written as “The system must....”
- Is generally stated in positive terms (i.e., "the system must xxxx" vs. "the system must not xxx"); however, there are times when "must not" is the more appropriate way to express the requirement.
- Is **measurable**; includes a **measure or metric** that can be used to determine whether the requirement is met (e.g., time or quantity), where appropriate; avoids the use of terms that cannot be defined and measured, such as "approximately," "robust," "user friendly," etc.
- **Is achievable and realistic; avoids terms such as "100% uptime," or "no failures."**
- Is complete; it can stand alone and be understood.
- Must be **testable**; that is, there must be some way to test the system to determine whether the requirement is met.

Below are some examples of poorly written and well-written requirements, with explanations of what is wrong with the poorly written requirements statements.

Poorly Written Requirement	What Is Wrong	Well-Written Requirement
Users must have access to their personal data, which will be transmitted in a secure manner.	Two requirements (in this case, one user and one system performance) are expressed; each statement should express only one requirement.	1. The system must provide a user with access to their personal data. 2. The system must transmit personal data in a secure manner.
The system must calculate the total of all items in the online or website shopping cart and display the total to the user.	Two requirements are expressed; each statement should express only one requirement.	1. The system must calculate the total of all items in the online or website shopping cart. 2. The system must display the total of all items in the online or website shopping cart to the user.
Report must be provided within 5 seconds of the user clicking on "submit."	Not a complete sentence; and should be stated as "The system must..."	The system must provide the report within 5 seconds of the user clicking on "submit."
The system should require the user to provide a shipping address.	Avoid the use of "should"; use "must."	The system must require the user to provide a shipping address.

Poorly Written Requirement	What Is Wrong	Well-Written Requirement
The system must be easy to use.	"Easy to use" is not measurable or testable.	The system must provide on-screen prompts to guide the user through the correct steps to place an order.

The Requirements Document

Once the requirements statements are written correctly, they should be grouped into categories. The first categorization is whether a requirement is essential or nice to have. As stated above, this is done by asking the individual who identified it as a requirement, rather than using the analyst's judgment. Then, the requirements are grouped by the function or process involved so that the user community can understand them. Using the accounting system example, the requirements might be grouped under headings like: accounts receivable, accounts payable, payroll processing, financial reports, etc. Arranging the requirements in a sequence that follows the steps in a task is also helpful. For example, in establishing a receivable account, there are specific steps taken; if the requirements are listed in the order that is generally used, it allows the end user to ascertain whether the list of requirements is complete and accurate. Each requirement statement will be assigned a unique identifier so that it can be referred to with ease and clarity. A full requirements document or "requirements specification" may contain many hundreds, or even thousands, of requirements. Again, more detailed requirements are needed for systems being built in-house or under contract. In the case of selecting a COTS product, only the higher level essential user requirements and the system performance requirements need to be developed. Otherwise, if too many specifics are identified, it may be impossible to find a COTS solution.

If all this documentation of requirements seems like it is very time-consuming, *it is!* Identifying and documenting the requirements is the basis upon which all further system decisions will be made, so it is a valuable investment of time and human resources. The later in the process that requirements changes are introduced, the more costly they become to implement. In developing a system, it would require the developers to go back and re-do portions of the system and re-test all the possible outcomes; and, depending on the severity and impact of the change, it may prove to be extremely costly. For COTS solutions, a significant change to one or more essential requirements may impact which systems should even be considered. The upfront investment in defining the requirements helps prevent downstream costs and delays.