Whether considering building, enhancing or buying a surveillance system, you must first take the key step of defining your requirements; that is, specifying what you need your system to do to support your work. Whether large or small, every information system needs to enable the work and workflows of the staff doing the work.

Requirements define the "what," not the "how," of information system functionality. You can trace back most frustrations with software to a lack of clear requirements or poor execution of those requirements.

This document provides practical advice on how to create and validate clear, concrete, actionable and meaningful system requirements.

The responsibility to have clear requirements rests with the program, not IT or a vendor; the program staff that does the work must own and understand their own business requirements.

Outline requirements by business process

The traditional way to generate and organize requirements is by *business process*, the set of activities and tasks that, once completed, accomplish an organizational goal. The table below provides examples of requirements for the surveillance business process "Data Analysis and Visualization." After determining your business processes, use the Toolkit's Requirements Template to define and document the associated system requirements.

ID	ACTIVITY	REQUIREMENT (The system must or should…)
1	Define report purpose	Allow user to select multiple pre-defined report types (i.e., situational awareness, outbreak management, ad hoc report, etc.)
2	Define report purpose	Allow user to create customized report types
3	Define report purpose	Allow user to view a list of commonly-used report purposes/questions
4	Define report purpose	Allow user to view similar reports that have been recently executed by other users
5	Define report purpose	Allow user to customize report (e.g., add titles, context, purpose and author)
6	Define report purpose	Support recognition of parameters stated in natural language
7	Define report purpose	Allow user to generate internal work flow reports (e.g., completeness of case investigations, number of contacts, etc.)
8	Define audience	Allow user to select audience type or group

ID	ACTIVITY	REQUIREMENT (The system must or should…)
9	Define audience	Allow for pre-defined audience types/groups
10	Define audience	Allow for user-defined types/groups
11	Define audience	Prompt user with questions to assist with defining audience
12	Define audience	Allow user to maintain contact information for report recipients (e.g., name, title, email, phone, language preference, etc.)
13	Define audience	Support ability to interface with contact management system
14	Define audience	Allow user to easily create, save and modify contact data
15	Define audience	Allow user to select audience level of education
16	Define parameters	Allow user to select multiple variables for analysis
17	Define parameters	Allow for pre-defined parameters
18	Define parameters	Allow user-defined variables (e.g., age = 1 year to 2 years+5days)
19	Define parameters	Support Boolean search logic
20	Define parameters	Allow user to save selected parameters for future use

Note that the template enables you to also document high, medium or low priority. This is important because system development/enhancement seldom involves everyone getting everything they want.

Benefits of requirements

Well-written requirements tend to offer:

- A higher return on technology investments
- A faster reaction to changing business situations
- Decreased frustration and stress for all involved
- Reduced misunderstanding in communication

- Fewer unneeded features
- Better relationships with your internal or external system developers
- Greater leverage when working with a central IS department/unit

Basic principles for defining requirements

Good requirements tend to follow some basic principles:

- **Be simple**: State one thing and state it well; don't use compound sentences (and, or, but), and avoid using limiting phrases.
- **Be complete**: Do not depend on other sources of information, and always include a subject, verb and object. For example, instead of writing "Submit an application," write, "Website visitor should be able to submit an application for health insurance coverage."
- **Be well-structured**: Identify an actor and describe what the actor should or should not do. For example, write, "Underwriting should process applications within one business day," rather than, "Applications must be processed by underwriting within one business day."
- **Emphasize *what*, not *how***: A good requirement emphasizes what should be done or what the results should be, not how to do it or obtain those results. To do this, a requirement needs to:
    - Avoid preconceived solutions.
    - Describe business logic (rules), not technology solution.
    - Express the "what" (destination), not the "how" (journey).

When defining requirements, remain particularly vigilant about the last point. Teams often default to conversations about how to do things rather than what needs to be done. For example, your team might come up with:

> *The order form should provide a drop down box of state abbreviations*

This requirement focuses more on *how* to capture particular information and doesn't include an actor. A better requirement would be:

> *The system should allow the customer to provide a valid state abbreviation*

By simply stating what you need the system to enable rather than the specific approach to providing a state abbreviation, you allow developers to determine or recommend options to enable this function. They may know of additional, perhaps better ways to accomplish a given requirement than your team may know.

For a video overview of functional requirements, go to **https://www.phii.org/phii-voices/functional-requirements**.

For an animated overview of defining requirements, using active surveillance as an example, see **http://phii.org/crdm/defining-requirements.html**

Validating your requirements

Once you've identified required and optional functional system requirements, you will want to validate them through one or more processes. Depending on your situation, you may use one or more of the methods described below. Regardless of the method you choose, be sure to involve all of your stakeholders, particularly users, and stop once you have achieved consensus.

Checklist for validation

Look through your requirements and verify that they are:

- Accurate
- Consistent
- Feasible
- Able to be validated
- Clear and simple to understand
- Numbered for reference
- A single requirement per statement; not multiple requirements lumped together

Desk checking

Desk checking involves asking individual stakeholders to conduct a written review of your requirements. When desk checking, a stakeholder:

1. Reads the document through without documenting defects
2. Determines and documents any missing major aspects
3. Determines and documents the clarity of the document structure
4. Reads through the checklist to remember particular points to consider
5. Rereads the document with the checklist in mind
6. Records defects and queries noticed
7. Sends the list to his or her business analyst and keeps a copy

Walkthrough

Walkthroughs are facilitated group reviews of the requirements. When conducting a walkthrough:

1. Gather a group (usually ten or fewer) to discuss the requirements and find defects
2. Ask the business analyst to be an observer and resource
3. Assign a neutral facilitator
4. "Walk through" the document page by page, looking for and recording defects

5. Use the checklist above and jot your thoughts
6. Have your facilitator or analyst make a list of defects and queries for document revision

Peer review

The peer review resembles a walkthrough; it is highly structured and follows a set of rules or guidelines. Use peer review when validating high-risk projects with the goal of lessening risks by identifying as many defects as possible.

Guidelines for peer review

Consider adhering to the following guidelines when conducting your peer review:

1. Define groups of four to five peers to represent various functional groups (i.e., create a peer group for users and a peer group for management).
2. Have each member:
    - Prepare for the peer review meeting by performing an individual review using a checklist
    - Bring a list of defects found to the meeting
    - Track preparation time
    - Assume one of the following specific facilitator roles during the meeting:
        - **Author**: makes sure people understand the intent of a requirement, makes revisions after the meeting.
        - **Facilitator**: sends out agenda and pre-meeting instructions, facilitates meeting, and makes sure revisions are made.
        - **Reader**: reads each requirement in turn to introduce it.
        - **Recorder**: takes notes on all problems identified; should indicate seriousness.
        - **Timer**: provides time checks at quarter, half and three-quarter meeting intervals.
3. Ensure reviews last no more than two hours including a break.
4. Focus on finding problems and only problems—not solutions.

Procedures for peer review

The following section lists typical procedures involved in a peer review, along with the role that will carry out each procedure:

1. Before the meeting:
    - Facilitator identifies participants and sends out requirements and supporting materials for review.
    - Facilitator schedules the meeting, taking into account time demands on participants.

2. During the meeting:
    - Facilitator asks the group for significant items missing from document.
    - Facilitator asks the group for broad comments about the document.
    - Reader examines or paraphrases each requirement (or set of requirements) and any associated comments.
    - Facilitator maintains the focus on identifying problems; limits discussions about solutions.
    - Recorder tracks problems noted for each requirement.
    - Timer informs the group when specific time intervals have passed.
3. After the meeting:
    - Author revises the document based on feedback gathered during the meeting.