

Chapter 4 Requirements Analysis

4-1. General

a. Geospatial Data Systems are successful only when they comprehensively and consistently meet the needs of users. As such, the procurement, installation, and use of a successful GDS depends on well-defined user requirements obtained through a development stage known as requirements analysis. Requirements analysis is a detailed study of the needs of potential users of the GDS which helps cut down the costs of acquiring and maintaining a GDS. The requirements obtained during this study may come from a variety of sources including system operators, domain experts (someone with an in-depth knowledge of the functional area), requirements and software engineers, managers, existing systems and standards, specifications, and any other potential users. When completed, the requirements analysis should result in a clear statement of end-product characteristics, required production rates, estimated data volumes, and a cost/benefit rationale.

b. The documentation required to justify and validate the acquisition of Automated Information System (AIS) hardware and/or software is currently in transition between the General Services Administration (GSA) and the Office of Management and Budget (OMB). It is anticipated that policy and direction down to MACOM level will not be received until the Fall, 1996. Project Managers should contact their local Director or Chief of Information Management for guidance and assistance in the required documentation. Experience has shown that a more rigorous requirements definition is needed for GDS.

4-2. What is a Systems Requirements Analysis?

a. This section defines a systems requirements analysis and explains many of the elements of a good requirements analysis. GD&S require data to operate, so a separate, but similar, data requirements analysis is needed to ensure that the system has the data sets needed to perform the specified functions. The data requirements are discussed in detail in Chapters 7 and 8 of this manual.

b. The earliest stages of the system development cycle should involve information gathering about the system to be procured. A committee determines the GD&S needs of each potential user through written, verbal and face-to-face contact. These needs or requirements can include anything from types of data the GDS can handle to necessary product output types. Once the requirements have been gathered and compiled into a manageable set, they form the basis for a

System Requirements Specification (SRS) which is used to determine all the external features of the GDS. Requirements analysis is the activity of understanding the application domain, the specific problems and needs of the users, and the constraints upon possible solutions. When a requirements analysis is done properly, it is possible to match the description contained in the SRS to several systems and pick the GDS which best suits all the users in the agency.

c. One of the misconceptions about requirements analysis is the "what versus how" dilemma. A requirements analysis study should only include any external behavior of the GDS, however, many people tend to confuse the external and internal behaviors and try to describe not only their needs but also how the system should solve those needs. The "how" aspect of the system should be left to the design phase of the development cycle which comes after the requirements analysis has been completed. GDS system requirements are capabilities needed by users to solve GDS problems (e.g., performing a site suitability analysis which involves graphical and textual data and multiple maps with differing projections), and these requirements must be met by a system component (e.g., computer system, software, data, etc.) in order to satisfy a contract, standard, or system specification. System requirements can be broken down into seven main categories:

- ∨ Functional requirements specify transformations that take place in the software system (i.e., the inputs and outputs for each function).

Example: The software shall identify all overshoots and undershoots by placing a box around the dangling end of each line segment.

- ∨ Behavioral requirements specify the way the system reacts to external and internal stimuli (i.e., what causes certain activities to start, stop, etc.).

Example: The program shall pause and return to the previous condition when the *escape* key on the keyboard has been depressed.

- ∨ Performance requirements specify timing, throughput, and capacity of the system.

Example: The system shall be able to return the geographic location of any address in under 3 seconds.

- ∨ Operational requirements specify how the system will run and communicate with its human users.

Example: The system shall provide a help facility, accessible from software, that describes the system components.

- Interface requirements specify characteristics of the human/computer interaction, such as screen layouts and validation of user data entry. In the case of system, an interface requirement can also specify hardware interfaces.

Example 1: Error messages must be displayed on the screen starting at row 1, character position 1.

Example 2: The system must be targeted to run on Sun workstations with the Solaris 2.1 Unix operating system.

- Quality Requirements specify system reliability, useability, and maintainability.

Example: The system must not fail more than 3 times during 1,000 CPU hours of operation.

- Constraints specify restrictions that impact the system in some way.

Example: Each line segment will be limited to 255 coordinate pairs.

d. In addition, there can also be security requirements when the system is going to be working with classified data and data requirements dealing with the validation and accuracy of the data sets passed to the system.

e. When the task of gathering the system requirements is over, it is important to sort through them to build a manageable set for the SRS. You will probably find that many requirements have been repeated several times although they may be worded differently. Others may be ambiguous, needing additional input from the source of the requirement. The Carnegie Mellon Software Engineering Institute has developed guidelines for generating such manageable sets. In these guidelines, they describe the seven requirements error types to watch out for when organizing the requirements set:

- System requirements are ambiguous if there is more than one interpretation. Ambiguity arises due to the pitfalls of natural language.

Example: Total item count is taken from the last record.

Problem: How do you interpret last record?

- System requirements are incorrect if some fact within the requirement has been misrepresented.

Example: For all computations, use the northwest corner as the origin of the data set.

Problem: What if the southwest corner were the origin for some of the data sets?

- System requirements are incomplete if one or more necessary facts (or requirements) have been omitted.

Example: Error messages must be displayed on row 24 of the terminal screen.

Problem: How will the system display multiple, simultaneous error messages?

- System requirements are inconsistent if they are in conflict.

Example: C must be computed as $A + B$
C must be computed as $A - B$

Problem: C has two definitions.

- System requirements are volatile if they are susceptible to change.

Example: The system shall handle only one data set at a time.

Problem: Although this may satisfy current needs, what happens if you need to work with multiple data sets simultaneously?

- System requirements are untestable if no cost-effective way exists to verify them.

Example: The system shall have a good user interface?

Problem: There is no way to define a "good" interface, so there is no way to verify it.

- System requirements are nonapplicable if they are not relevant to the problem.

Example: The operator of the system must be standing upright during operation of the system.

Problem: This requirement has nothing to do with the development or procurement of the system.

f. These guidelines should help throughout the systems requirements analysis process. The requirements analysis is described from start to finish in Chapter 4. Remember, that once you have formulated an SRS from your requirements

set, it should be used as a “bible” from which a GDS will be procured. The SRS can help identify potential errors before they become costly to fix, so you should treat this stage in the GDS life cycle with the utmost importance.

4-3. Why is Requirements Analysis Important?

A requirements analysis can appear to be an overwhelming task to complete for a Command-wide GD&S procurement. However, the benefits of performing a comprehensive analysis prior to system acquisition are well-documented and easily justify the effort.

Errors are much easier and less costly to correct when they are detected early in the development stage rather than later on. The reason for this is that one component in a system will often build upon the capabilities of another until you have a fully-integrated system. In any failed system development, it can be demonstrated that when one component is missing certain required capabilities, this problem will propagate itself through the rest of the system. Other components will not be able to meet certain needs, and so on with a cumulative effect on the entire system. This lesson is equally true for an “off-the-shelf” GDS and a custom-developed GDS.

The document most often used to record the system requirements is the System Requirements Specification (SRS). It clearly explains every detail of the system without ambiguity or error, so that the buyer can make intelligent choices about the system to be procured. In addition, this document also serves as contract between the buyer and developer/supplier; the GDS is not complete until the developer/supplier has met every requirement contained in the SRS.

4-4. Who Should Perform the Requirements Analysis?

In “A Process for Evaluating Geographic Information Systems,” Guptill writes:

The first problem faced by any organization considering the implementation of a [GD&S] is to determine who should perform the requirements analysis. Requirements Analysis' for successful [GD&S] installations have been performed by in-house staff, contractor staff, or through a combination of both approaches. There are many valid reasons for opting for any given approach, however, the desired result is the same, to develop a comprehensive assessment of the analytical capabilities and products required by potential [GD&S] users. The requirements of the users can then be matched with system capability to determine optimal configurations for the organization's [GD&S] procurement.

In-house staff inherently have a greater understanding of the tasks which are to be considered for automation through [GD&S] technology. This unique knowledge may justify training staff members in [GD&S] technology and requirements analysis techniques so that the requirements analysis may be performed in-house. In cases where existing staff members have expertise in GD&S's there may be little reason to consider bringing in outside assistance.

When staff time or skills are not available, or when new programs and concepts are being proposed that staff is not experienced with, outside resources may be required to perform the requirements analysis. Assistance may also be available from resources within the parent agency of the organization considering the [GD&S] implementation. Assistance in developing Requests for Proposals (RFPs) for requirements analysis services may similarly be available within the agency.

The important element in determining who should perform the requirements analysis is assuring that the provider of the service has a thorough understanding of both [GD&S] technology and the operations of the organization. When an outside organization is brought in to perform the requirements analysis, it is the responsibility of the technical representative or point of contact for the requirements analysis services procurement to assure that the contractor fully understands the organization's products, services, missions, and needs.

Possible conflicts of interest should also be considered before a final determination is made as to who should perform the requirements analysis. Organizations and individuals may, in some instances, have a vested interest in certain hardware or software types, and may be inclined, whether intentionally or unintentionally, to bias the results of the requirements analysis toward particular systems. The objective of the requirements analysis is to identify the needs of an organization and then to select the [GD&S] that best fits those needs, if such a system exists. All reasonable effort must be made to assure this goal is realized, including assessing possible conflict of interests, or biases, on the part of persons or organizations that potentially could perform the requirements analysis.

4-5. Performing a Requirements Analysis

The purpose of this section is to lay out a generic framework for performing a requirements analysis with some key tips where appropriate.

a. *Elicitation.* The first step in performing a requirements analysis is to obtain system requirements from all persons involved with the use of the GD&S system. This

step, known as requirements elicitation, can involve any of the following activities:

- ∩ Interviewing.
 - Face-to-face contact with potential users, manager, etc.
 - Should identify relevant positions from a formal organizational chart.
 - Identify the work flow interactions between users and the rest of the organization.
 - Ask context free questions such as:
 - “Who else should I talk to?”
 - “Who else may use the system?”
 - “Who else interacts with you?”
 - Inform interview candidates ahead of time and give them any relevant material.
 - Secure an adequate time commitment from candidates.
 - Give the person reasonable courtesies in terms of answering your questions.
 - Periodically confirm your understanding of the person's responses.
 - Summarize your understanding at the conclusion of the interview.
- ∩ Brainstorming.
 - A simple technique for generating ideas.
 - Can be used for generating alternate viewpoints of the problem.
 - Works best with groups of 4-10 people.
 - Outcome depends on the expertise and knowledge base of the participants.
 - Generation phase: a leader provides a seed expression to the problem.
 - Generation phase: participants freely generate ideas relevant to the problem.

- Generation phase: all ideas are placed on a large board or sheets of paper.
- Generation phase: should be stopped when ideas become low (~ 15-20 min.).
- Consolidation phase: ideas evaluated, outliers removed, related ideas combined.
- Consolidation phase: remaining ideas are grouped and classified.
- ∩ Scenario generation.
 - Determine needs through real world system usage scenarios.
- ∩ Rapid prototyping.
 - Quick user interface or basic system shell construction.
- ∩ Modeling.
 - System portrayal through means such as data flow diagrams.

b. Analysis. The next step in the process is to organize the information received from the elicitation process. At this stage you want to remove any requirements errors you can find by looking for ambiguities, inconsistencies, and any of the other seven requirements errors previously mentioned. You also want to remove any requirements which are duplicates. The main goal here is to build an organized set of requirements that clearly state the system and its behavior with as few requirements as possible. If additional information is required, you should go back to the elicitation stage before continuing, because you will need the final requirements set before determining feasibility.

c. Feasibility. It may or may not be possible to procure a system with all the capabilities requested by users, so you must next perform a feasibility analysis. During this phase, it is a good idea to conceptually plan the system without getting into too much detailed design. You should also speak to representatives of commercial GDS vendors to discuss your requirements and how their systems could be used to meet them. System modeling through data flow diagrams, decision tables and trees, and control flow diagrams is a good way to visualize the system from the given requirements set. Matching the capabilities of possible GDS solutions against these models will give you an understanding of what will be required to build the necessary GDS, either as a custom software development or as a customization of a commercial GDS. If specific

requirements are making the GDS too expensive or risky to build, you must decide whether or not to write the SRS and have the system developed as-is or to eliminate the infeasible requirements.

d. Specification. Without an SRS, the contractor/in-house development team would have no idea of what was to be built, your users would be left with false expectations of the GDS, and there would be no way to test if the solution meets all the needs captured during the elicitation and analysis phases. Therefore, you must document the external behavior and qualities of the system while excluding any internal information concerning how the software and hardware operates and the algorithms it uses. The SRS has four roles in the development life cycle: (1) it is the primary input to the design team; (2) it is the primary input to the system test planners; (3) it controls the evolution of the system; and (4) it communicates an understanding of the system requirements. The SRS should be understandable to anyone who reads it regardless of their background, and should contain the following qualities:

- ∨ An SRS is correct if every requirement stated therein helps to satisfy a user need.
- ∨ An SRS is complete if every user need is satisfied by a system that satisfies every requirement in the SRS.
- ∨ An SRS is unambiguous if every requirement stated therein has only one possible interpretation.
- ∨ An SRS is consistent if no subsets of the requirements stated therein conflict.
- ∨ An SRS is logically closed if it specifies a response for every conceivable stimulus in every conceivable state.
- ∨ An SRS is organized if its readers can easily locate information.
- ∨ An SRS is modifiable if it can be easily changed when errors are found, or when it needs to be changed due to changes in requirements.
- ∨ An SRS is traced if the origin of each of its requirements is clear.
- ∨ An SRS is traceable if other documents can reference its requirements easily.
- ∨ An SRS is concise if it cannot be made shorter without jeopardizing other qualities of the SRS.

- ∨ An SRS is verifiable if there exists a finite, cost effective technique to check that each requirement in the SRS is satisfied by a system.
- ∨ An SRS is annotated by importance if the relative importance of each requirement is indicated.

4-6. Performing a Data Requirements Analysis

For most GD&S, the data are the largest cost component, often eclipsing the cost of the hardware and software combined. It is critical that they be properly managed from initial design to long-term archive. The data requirements analysis is the first step in building or acquiring a geospatial database. It is used to both define the content and format of the data as well as to inform potential users of the intent to acquire/produce the database. It is conducted similarly to the system requirements analysis, with the objective of establishing the minimum set of requirements that will meet the near- and long-term needs of the users. It should establish the database:

- ∨ Structure.
- ∨ Content.
- ∨ Scale/resolution.
- ∨ Accuracy.
- ∨ Currency.
- ∨ Any special user requirements.

a. Structure. The structure of the database defines its basic makeup. The two most common geospatial database structures are vector, such as road networks or engineering drawings, and raster, such as gridded elevation matrices or digital imagery.

The database structure is established by the application(s) of the database and the systems that will be used. Applications that involve detailed analysis of lineated features, such as road or stream networks, require attributed vector databases. Raster databases are ideally suited to applications that involve the boolean combination of areal, layered information. The computation of cross-country mobility, which is a function of ground slope, surface roughness, vegetation coverage, and other terrain characteristics, is an example of a query that is often performed using raster databases. The structure of the database should be determined very early in the requirements analysis.

b. Content. The content of the database is determined by the applications for which it will be used. If the database

has a vector structure, then it is necessary to determine the specific inclusion criteria for features and attributes. For example, if roads are required, to what level (highway versus dirt cart path), and how richly must they be attributed (e.g., surface material, number of lanes, whether the road has a median, whether the road access is controlled, whether the road is one-way or two-way, etc.)? Collecting unnecessary features and attributes adds to the cost of the database, so it is important to establish the minimum set of features and attributes that satisfy all user requirements.

If the database has a raster structure, then the thematic layers must be determined. Raster databases may have only one layer, such as a digital image, or it may have many layers, such as those needed for the cross-country example discussed above. As with vector databases, increasing the information carried in a raster database increases the cost of production, distribution, and archive, so it is important to establish the user requirements early in the design.

c. Scale/Resolution. Scale refers to the collection scale of vector databases. A collection scale of 1:24,000, for example, implies that the vector database has a feature content that is roughly equivalent to a hardcopy USGS 7.5-minute Quadrangle. Resolution refers to the size of the pixels in the layer(s). Resolution may be expressed as a distance on the ground (e.g., 10 meters) or as is common for raster databases produced by scanning hardcopy products, pixels per inch (e.g., 300 dpi). The scale/resolution has a significant effect on the ability to perform certain types of queries; a database with insufficient scale/resolution may preclude analysis at the level of detail required, but a database with an excessive scale/resolution is more expensive to build and has a negative impact on storage volume and processing times.

d. Accuracy. The accuracy of a database refers to several factors, including coordinate accuracy, attribute accuracy, logical consistency, and completeness.

(1) Coordinate accuracy. Coordinate accuracy refers to the accuracy, expressed as a distance and a confidence factor, of the geographic positions in a database. The coordinate accuracy of a database is influenced by, but not controlled by, the scale/resolution.

(2) Attribute accuracy. The attribute accuracy reflects the confidence in the codes and attribute values assigned to features in a vector database.

(3) Logical consistency. Logical consistency is a measure of the relative positioning of features in a vector database. An example of a break in logical consistency is a building captured on the wrong side of a road. While both

the road and the building may be within their allowed coordinate accuracies, their relative positioning is not consistent. Logical consistency is most often expressed as the percentage of features that are consistent with all surrounding features. SDTS expands this definition to include the general fidelity of the data capture, considering such factors as the presence of overshoots and undershoots, topological integrity, and graphic presentation.

(4) Completeness. Completeness refers to the percentage of features in the real world that are captured in the database, within the capture rules of the particular database. Completeness, which deteriorates over time as new features are added to the real world, is measured at the time of database construction.

e. Currency. The currency of the database refers to the elapsed time since collection of the source material. Areas change at different rates, e.g., cultural features in suburban areas change much more rapidly than in sparsely populated regions, so currency requirements for databases often vary according to the area being captured.

f. Special user requirements. This section of the requirements analysis captures any unique needs for the database. These might include: on-line access to the database by the users, unique archive requirements, or special services such as on-demand datum or coordinate transformations.