# THE REQUIREMENTS GENERATION SYSTEM: A TOOL FOR MANAGING MISSION REQUIREMENTS

**Sylvia B. Sheppard**
**NASA Goddard Space Flight Center**
**Greenbelt, MD 20771**

## ABSTRACT

Historically, NASA's cost for developing mission requirements has been a significant part of a mission's budget. Large amounts of time have been allocated in mission schedules for the development and review of requirements by the many groups who are associated with a mission. Additionally, tracing requirements from a current document to a parent document has been time-consuming and costly. The Requirements Generation System (RGS) is a computer-supported cooperative-work tool that assists mission developers in the online creation, review, editing, tracing, and approval of mission requirements as well as in the production of requirements documents. This paper describes the RGS and discusses some lessons learned during its development.

## INTRODUCTION

One of the most important, time-consuming and expensive tasks for any mission is the development of mission requirements. For example, consider the contractor time expended for development of requirements for the Payload Operations Control Center (POCC) and the Command Management System (CMS) portions of two Small Explorer missions. For the Fast Auroral Snapshot Explorer (FAST) and Submillimeter Wave Astronomy Satellite (SWAS), the contractor person-years expended were 10.7 and 8.0, respectively (Mandl, 6/9/94). (The data do not include civil service time or time expended on requirements for other parts of these missions.) Similar expenditures for the Xray Timing Explorer (XTE) were estimated at between 15 and 18 person-years.

The Requirements Generation System (RGS) was developed to help automate the requirements process. The goal was to reduce mission schedules and costs associated with the creation and use of mission requirements information. We hypothesized that we could meet this goal by:

- increasing communication about all levels of mission requirements among the many individuals and groups of personnel contributing to a mission,
- providing automated assistance for the online development, editing, review, tracing and approval of requirements and the production of related documents and reports, and
- reusing sets of requirements across similar missions.

## RGS CAPABILITIES

The RGS uses a distributed system architecture to encourage online work. The RGS was designed for existing desk-top platforms (i.e., Macintoshes and PCs). The sections below present the operations concept.

## Single Mission Database

The RGS uses a single mission database to maximize communication among mission personnel and to facilitate the online management of mission information. The mission-specific database is made available to all mission personnel from the beginning of a mission throughout its life cycle.

Figure 1 shows the mission requirements documents that are produced for a standard mission. The letters above each box list the organization (or person) responsible for producing the document. All levels of requirements and all requirements documents, commentary, and rationale are consolidated in the RGS mission database, thus reducing the time to locate, review and disseminate information. Higher-level requirements, such as those found in the Detailed Mission Requirements (DMR) document, and lower-level requirements, such as those found in the System Requirements Document (SRD), are included. This eliminates the need for traceability across separate documents and/or databases and allows for the production of reports that contain requirements at varying levels of detail. Additional documents (e.g., the Mission Requirements Request) are available online for reference and for the explicit traceability of DMR requirements to requirements in parent documents.

Although working online with the RGS does not eliminate the need for meetings to discuss issues, it can reduce the time needed to agree on a set of requirements. Mission personnel no longer need to wait for the release of a document to review requirements; they can be reviewed, and approved or rejected, individually. Piecemeal review can result in schedule efficiencies.

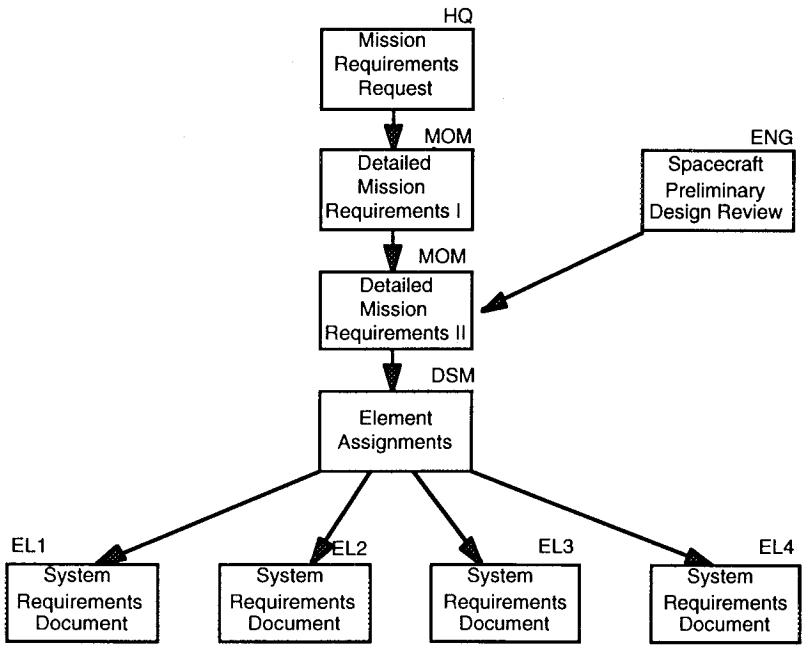


Figure 1. Requirements Document Hierarchy

## Online Entry and Editing

The RGS provides a form-based graphical user interface for entering and editing requirements on-line. Requirements may be entered in any order. They are numbered by the requirements developer as they are entered and may be hierarchical. In Figure 2 the first requirement is numbered 4100-1, the second, 4100-1.1, etc. In this example the section of the requirements document (i.e., 4100) is appended to the beginning of the requirement number. This addition is optional.

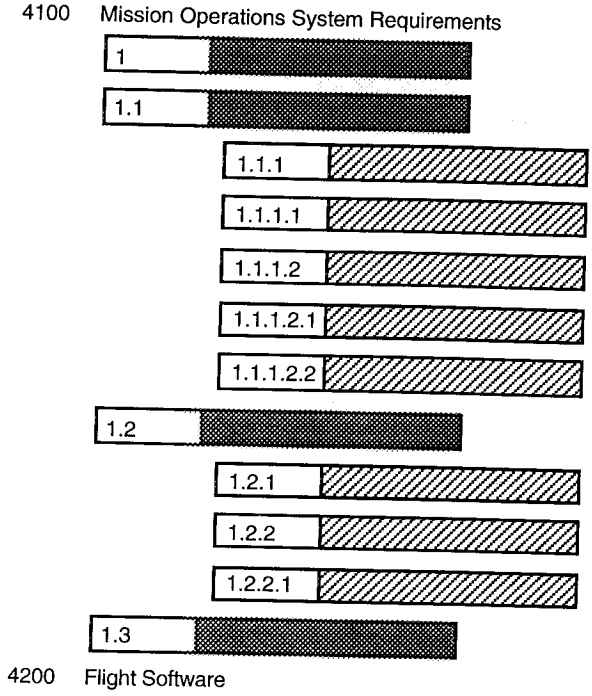


Figure 2 : Sample Requirements Hierarchy

Users may assign a "level" to each requirement. This number associates the requirement with a given degree of detail or a given document type (e.g., the requirements in the Mission Requirements Request might be designated Level 1, the Detailed Mission Requirements document Level 2, and the System Requirements Document Level 3). The requirement level is independent of the hierarchical number assigned to individual requirements. Mission personnel may determine the relationship between the levels and the hierarchy for their mission. Figure 2 shows one possible assignment.

A table of user privileges defines which users may enter and edit which sections and levels of the requirements. For example, the Mission Operations Manager (MOM) might assign the privilege of entering/editing the Level 2 requirements of Section 4000 to one group of requirements developers; Level 3 requirements for Section 4000 might be assigned to a different group of requirements developers.

## On-line Approval/ Rejection

The RGS provides on-line, form-based capabilities for appropriate mission personnel to approve or reject a requirement and to attach associated rationale for their decisions. Approval privileges for a specific level of requirement may be assigned to any of the mission personnel. For example, for Level 2 requirements, approval could be assigned to only the MOM; to the MOM and Data System Manager (DSM); to the MOM, DSM and Element Manager (EM); or to some other combination of mission personnel. Different EMs may be assigned approval privileges for different sections of requirements as appropriate.

The RGS annotates each mission requirement in the database with a "status" that describes how far the requirement has progressed toward final approval. A clearly labeled status field distinguishes work-in-progress requirements from mission-approved requirements (Table 1). The instant availability of newly-developed requirements (i.e., draft or pending) provides access to the current thinking on issues and allows for speedier review and response from interested parties. Further, approving requirements individually (as opposed to waiting for the release of a set of requirements in a document) can speed up planning and design. Finally, the overall view of the status of the requirements aids management in their assessment of the progress that has been made at any point in time. Inadequate requirements in a certain mission area can be identified, and measures can be taken to correct any difficulties.

Table 1. Status Classifications for Requirements

| | |
|---|---|
| Private | A requirement that is work in progress, visible only to the author (or the working group to which the author belongs). |
| Draft | A requirement that is work in progress, visible to anyone with access to the mission. |
| Pending | A requirement that has been submitted for approval. This requirement is considered "finished" but not accepted. |
| In Acceptance | A requirement that has been accepted by one, but not all of the parties responsible for approving the requirement. |
| Accepted | A requirement that has been accepted by each of the parties responsible for its approval. |
| Rejected | A requirement that has been rejected by at least one of the parties responsible for its approval. |
| Accepted with Contingencies | A requirement that has been accepted by each of the parties responsible for its approval, but to which the DSM has responded with exceptions. |

### On-line or Paper-Based Review and Reporting

Any requirement in the mission database may be reviewed by any mission user who has been granted privileges to access the database. An easy-to-use search mechanism allows users to filter the database and to select reduced sets of requirements for review. The selected requirements may be reviewed on-line or printed. Report contents can be defined by users using a simple selection technique.

Reviewers are also afforded an on-line "notes" capability for attaching commentary to individual requirements. The notes are then available for perusal by all database users.

## HARDWARE AND SOFTWARE ENVIRONMENT

The RGS has a client-server architecture. A client-server architecture uses client machine(s) and server machine(s), along with the underlying operating system and interprocess communication systems, to form a composite system that allows the distributed access, management, analysis, and presentation of information.

The RGS supports both PC and Macintosh computers as client machines. Future plans include running on UNIX platforms. A Compaq System Pro/LT comprises the server portion of the RGS hardware configuration. This server houses all the RGS databases, support documentation, and database software.

The RGS server resides on the GSFC Center Network Environment (CNE). GSFC users access the RGS server from their workstations via this network. Local off-site users access the CNE via a T1 line, while off-site users not local to GSFC access the CNE via the Program Support Communications Network (PSCN) Internet.

The RGS was developed using two Commercial Off The Shelf (COTS) software packages, OMNIS 7 and SQL Server. OMNIS 7,

manufactured by Blyth Software, is a graphical user interface package that was used to develop the front-end portion of the RGS. The front-end executes on the client machines and provides the mechanism for users to interface with the RGS database. The front-end is responsible for soliciting queries or directions from the user for purposes of data update, analysis and retrieval and for presenting the results of queries and commands to the user. The front-end may also perform data analysis on the query results returned from the server.

SQL Server, a relational database management system marketed by Microsoft, Inc., was used to develop the RGS databases. The functions of this server component of the RGS custom software are to respond to user queries issued by the client machines and to manage the RGS requirements databases and document library.

## DISCUSSION

The primary goal in developing the RGS was to produce a system that improves the development of mission operations. To date the RGS seems to be fulfilling that goal. The system is currently being used for five missions, and there are plans to use it on another six missions. Estimates are that requirements costs will be cut at least 50%, with even larger savings for missions that are similar enough to reuse major portions of the mission databases (Mandl, 1994).

A second reason for developing the RGS was to learn about client-server, computer-supported-cooperative-work (CSCW) systems. This section discusses some lessons learned from development of the RGS.

### Lesson 1: Plan on Becoming a Full Service Organization

Computer-supported cooperative-work systems are only useful when there is a critical mass of users. For example, the major hurdle in using an electronic meeting scheduler is that all participants must be able to read and respond or the scheduling activity is hampered (Grudin, 1990). Similarly, deployment of the RGS as a CSCW system would be useless if mission personnel couldn't access requirements online.

Providing physical access to the RGS for all mission personnel required more resources than originally anticipated. Originally, the RGS team had expected to supply the RGS application, the COTS packages (OMNIS and SQL Server), training, a user's guide, and an RGS hot line service for responding to questions. Additionally, we planned to provide for maintenance of the server and the centralized database for each mission. We later determined that we had to become a "full service" organization. Many of the end users, spread across the Center and beyond, did not have the expertise to acquire and install the software to run a client-server system. Client-server software is more difficult to install than software packages that reside on an individual workstation. Generally Macintosh installations generally were done quickly, but PC installations often required extensive analysis. Sometimes it took several hours to get the RGS installed. Conflicts with existing mail and other resident user packages were the rule as opposed to the exception.

Additional chores included wiring offices to get users networked to the CNE, and supplying and installing communications software and Ethernet cards. In the case of off-site contractors who did not access the RGS server through the Internet, we provided expertise in dealing with the telephone company to obtain communication lines.

The extra services were time-intensive and expensive. Had we not had resources to expend for these tasks, the whole project might have failed.

## Lesson 2: Employ Users' Groups with Full Representation

Working with users' groups is an integral part of the methodology of the client-server development team. We established an RGS Users' Group at the beginning of the project and met monthly thereafter to determine the requirements for the system. The Users' Group discussed the types of users and the capabilities each would need to do the mission's work. In some cases we used detailed scenarios of the tasks to be done by the individual types of users in order to determine that we correctly understood the requirements.

The RGS Users' Group was very helpful in defining requirement and developing a design. However, one type of user, the Data System Manager, was not represented in the beginning. Users who were present were not able to represent adequately the functions needed by the DSM. For a later release of the RGS we redesigned several features to include those functions. We concluded that an efficient development methodology for CSCW projects absolutely requires the active involvement of every type of user, regardless of their amount or type of use.

## Lesson 3: Design for Flexibility

One original goal of the RGS was to design a system specifically tailored to handle the Goddard Mission Operation and Data System Directorate's method of developing and managing mission requirements. The reasoning was that mission personnel were accustomed to a largely paper-based process, and convincing them to adopt an automated system could be done best by making as many of the elements of the automated process as familiar as possible. Other requirements systems that were reviewed did not provide the specific kinds of functions that are needed to satisfy the Mission Operation and Data System Directorate process. To this end the RGS developers and potential users worked together to define the user types and the functions each would perform. The RGS team incorporated those functions into the design.

The first two releases of the RGS were successful because of this approach. However, further discussions, often with the users' groups, highlighted the need for differences in capabilities from mission to mission. Examples include the desire to change the privileges of users and user types, to use the RGS for different types of documents (as opposed to the DMR for which it was originally designed), to tailor the approval processes for individual missions, to create different document structures and formats, and to create traceability to a wider range of parent documents. In short, the original description of RGS capabilities was well-defined and rather rigid. As more users became involved, they requested more flexible capabilities to suit their mission's style of operating.

One approach to making the needed modifications would be to hand-tailor the RGS software for each mission. This approach was rejected because of the inherent software maintenance costs, the problems of managing multiple versions of the same software, and the limitations of what can be changed in software with a short turnaround time.

Instead we chose to design generic capabilities. The latest version of the RGS has a flexible set of functions that can be tailored to a particular mission's need by the mission personnel. Beginning with Release 3, mission personnel may configure the RGS, without assistance from the developers, to allow the definition of any of the following:

- any number of mission-specific user types (e.g., a requirements developer, MOM, DSM, EM, read-only user, and other mission-defined users).
- a mission-specific structure for a requirements document (showing what sections are to be included).

- mission-specific requirement levels, allowing for a greater level of detail beyond the standard three document levels (i.e., MRR, DMR and SRD).
- mission-specific acceptance privileges (determining what approvals are necessary for what sections and levels of requirements).

These flexible capabilities made the RGS a more general purpose tool. We also expect them to reduce the software maintenance required for the RGS.

### Lesson 4: Plan to Deal with Changes in Work Flow

In the past, mission personnel developed requirements in a sequential fashion, largely completing and approving higher-level requirements documents before lower-level requirements were defined. Often one group of requirements developers wrote higher-level requirements and another group lower-level ones.

Use of an open database promotes changes to this traditional work flow. Requirements at any level can be entered at any time. Mission personnel can add information as soon as it becomes available. One section of the requirements can be completed at the lowest level before another section is begun at a higher level. Additionally, documents per se become less important. Requirements can now be reviewed and approved individually, or in sections, as opposed to at the "document" level. While these changes can impact the schedule positively, this flexibility may be upsetting to team members who are used to a more structured process. Managers need to be prepared to establish procedures to deal with the changing work arrangements that ensue from automation of this type.

## REFERENCES

Grudin, J. (1990). Groupware and cooperative work: Problems and prospects. In B. Laurel (Ed.), *The art of human-computer interface design* (pp. 171-185). Reading, MA: Addison-Wesley.

Mandl, D. (1994, June 9). *Cost (manyrs) to gather requirements for MOC's with and without RGS and SDT methodology.* Informal communication.

## ACKNOWLEDGMENTS

The author would like to thank the many people who contributed to the development of the RGS. Mark Stephens is the RGS Project Manager; Lisa Dallas served in that capacity before the birth of the twins. The two software development teams are Linda Cingel, Rachel Campbell, Lou Fenichel and Barbara Wrathall of Computer Based Technology Incorporated and Gary Chatters, Phillip Wolf and Teresa Bleser of Century Computing. Special thanks go to William Guion, Randy Harbaugh, Richard Harris and William Macoughtry for their encouragement and support.