Maintaining user participation throughout the systems development cycle

by RANDY J. RAYNOR
and
LINDA D. SPECKMANN
Texas Instruments
Dallas, Texas

ABSTRACT

Effective user participation is well known to be an important aspect of good system development methodology. Specific tools and techniques for managing user participation in all phases of the systems development life cycle are illustrated by a large business system development project at Texas Instruments. Emphasis is placed on maintaining a constant level of communication between user and developer as the system design evolves.
INTRODUCTION

During the past few years there has been an increased awareness of the importance of user involvement in every phase of systems development. The approaches to achieving effective involvement vary from new tools usable by the end user to full-time assignment of a user to assist in the development process. Most of these techniques have some drawbacks because the right kind of tools and personnel are not available.

Many of the tools require either extensive training time or expensive application software. The user assigned to a full-time systems job is too often one who is not vital to the business operations. This user may not be the best source of information for the system developers and, because of detachment from the project, has no direct authority to implement the completed system in the project.

This paper draws from recent experiences at Texas Instruments with the design of a large product configuration management system to illustrate the selection of a requirements definition methodology easily understandable by the user community, as well as techniques for continued user participation throughout detail design, construction, and testing. A brief overview of the initial phases of the systems development life cycle is presented, and additional information is cited from a prior paper. The focus of this paper is on the later phases of the life cycle. The tools and techniques outlined require little training and no application software, allowing easy implementation in other environments.

Case Study Background

The need for a new configuration management system was widely recognized at Texas Instruments, especially for the highly complex radar and guidance products developed for the Department of Defense. The development was chartered as a part of the Engineering Information System—an implementation of IBM's Administrative Engineering Information Management System (AEIMS) concept.

At Texas Instruments, development of new products is organized through projects operating like small companies, with a staff for engineering, drafting, and configuration management as well as other design support functional areas. There was no companywide configuration management organization or focal point through which to coordinate development of a functional requirements specification. Rather, there were a large number of independent configuration managers with an equally large number of approaches to product configuration management. Although some small systems did support configuration management functions, they were not widely used.

The configuration management system was required to control and report product baseline evolution from concept formulation through production sustaining, including product drawing and specification development, change approval and tracking, data management, product as-built verification, and logistics support. Thus, this system would affect not just those independent configuration managers, but also every other discipline involved in product development and manufacturing.

THE SPECIFICATION APPROACH

Historically, systems developers have often considered user involvement a necessary evil. With the configuration management system it is assumed that the user is the subject matter expert, the user's involvement and support in every phase of system development is required, the user's time is a valuable commodity, and without the user's commitment the system will probably fail. This assumption places a heavy responsibility on the team of users (which must effectively represent all users), and it explains the importance of techniques for efficient use of team resources.

During the initial analysis of the configuration management system and team identification, it became evident that the players had widely differing opinions on what configuration management was and how it should be accomplished. This problem prompted an approach to system specification different from that typically followed.

System specification was broken into three distinct phases: concept definition, functional requirements, and detail design. The concept definition specification identified at a high level how configuration management should be accomplished. The functional requirements specification identified in more detail what functional steps and data were required to achieve the basic operations of the concept definition. Neither of these two specifications made reference to the computer systems solution. The detail design specification then identified which functions should be computerized and gave the exact definition of the system design from the user's perspective.

Selection of a Specification Methodology

The selection of a system specification tool was based on the following requirements. It had to

1. Be easy for users to learn, easy to use, and easy to change.
2. Serve as an effective communication tool between systems analysts and users.
3. Support the top-down decomposition of concept/functional/detial design specifications.
DeMarco's data flow diagrams were selected. They had only a few simple syntax rules, which could be learned in an hour. The 8½ × 11-inch size of each diagram was also convenient for team review because an overhead projector could be used.

Figure 1 represents the use of data flow diagrams to support the decomposition from the concept specification to the design specification. The concept specification defined the top three to five levels of the data flow hierarchy. The functional specification added an additional one to three levels to the data flow hierarchy where required, until specific user tasks were identified. Then a functional task description was developed for each. The design specification mapped the functional tasks to system modules and defined the design of these modules.

CONCEPT DEVELOPMENT AND REVIEW

The concept definition document defined the fundamentals of configuration management and the necessary procedures to accomplish them effectively. The following topics were covered:

1. Concept Definition Overview—The configuration management development history and the purpose, scope, and introduction of the concept definition
2. Introduction to Configuration Management Concepts and Definitions—A definition of the major configuration management functions
3. Present Operations—A discussion of systems and procedures in each area
4. Data Flow Diagrams—The diagrams depicting the product life cycle
5. Benefits—A discussion of expected benefits from the implementation of a system
6. Appendix—A list of key users, discussion of existing systems, and a keyword glossary

Twenty members of the user community were selected by a company vice-president to participate in the review of the concept definition. The team was composed of representatives from 10 functional areas, with emphasis on configuration management, quality assurance, logistics, engineering, and manufacturing.

To use the large team in the best way, the concept definition data flow diagrams were divided for review so that each set had a primary effect on one functional group. Subteams of about six members were formed, each led by an expert in the functional subject matter.

The product of the review was a 73-page document that described the overall approach to the way configuration management should be performed throughout the life cycle of a TI product. More important, however, the result of the review was a good working relationship between developer and user, based on mutual respect, and an appreciation by users and management for the effort that would be required over the next several years.

FUNCTIONAL REQUIREMENTS DEVELOPMENT AND REVIEW

The objective of the functional requirements document was to record data flow and process requirements down to the task level. The specification addressed the following topics:

1. Functional Requirements Specification Overview—The objective and scope of the functional requirements document
2. Configuration Management Overview—A brief discussion of the fundamentals as defined in the concept definition
3. Configuration Management Subsystem Descriptions—An overview, set of data flow diagrams, and function descriptions for eight major areas
4. Configuration Management System Development
Methodology—An explanation of the development life cycle

5. Appendix
   a. Data flow diagrams and descriptions
   b. Survey trip reports
   c. Logical database views
   d. Data element glossary
   e. Keyword glossary
   f. Configuration management forms
   g. Review team minutes

The draft functional requirements specification defined additional levels of data flow diagrams below those of the concept definition, emphasizing areas for potential system application. Task descriptions were then developed to define further many of the processes in the lower levels of the data flow diagram. The task descriptions defined an operation in terms of the input and output data and a set of processing steps.

The functional specification review team was partially selected from the membership of the concept review team. Those retained were experts in their fields and had schedules that permitted participation. Other new members were added to strengthen representation from functional areas that were greatly affected by configuration management tasks. This review demanded a more detailed understanding of data associated with daily tasks and therefore required representation from the users with a working knowledge of the function.

The result of this review was an 1100-page document that defined the complete set of tasks required to accomplish configuration management. From this list, tasks could be selected for detail design.

Since the scope defined by the functional requirements specification would require 3 to 4 years to develop, an initial phase was defined for detail design and construction. The data flow diagrams developed in the functional requirements specification played an important role in the first phase of detail design as well as subsequent phases. They provided the complete set of requirements used to prioritize later system releases.

The relatively small scope of the initial phase not only provided near-term system startup, but also improved user involvement and motivation by providing a goal requiring project level planning for implementation. The phased schedule is illustrated in Figure 2.

DETAIL DESIGN SPECIFICATION

The detail design specification defined the user interface to the application system, procedures for use of the system, overall software architecture, and database design. The contents of the detail design specification were as follows:

1. Introduction—The purpose and scope of the detail design specification document
2. System Description—An overview of the entire system's functional design
3. Subsystem Description—Data flow diagrams of all system functions, procedural flows and narratives, and online transaction descriptions
4. Changes to Existing Procedures—A step-by-step procedure describing manual and system operations
5. Test Plan—Test objectives and methodology
6. Training Plan
7. Appendix—Logical database design, manual forms, review team minutes, and glossary

Because of the detail required in the design specification, data flow diagrams were insufficient to represent the required information. The strength of the data flow diagram is its hierarchical illustration of a major problem. When a detail design is being made, procedural flows, control points, feedback loops, and organizational responsibilities become important. Therefore, procedural flows were chosen to show how the system would be used in day-to-day operations. However, as

---

The draft functional requirements specification defined additional levels of data flow diagrams below those of the concept definition, emphasizing areas for potential system application. Task descriptions were then developed to define further many of the processes in the lower levels of the data flow diagram. The task descriptions defined an operation in terms of the input and output data and a set of processing steps.

The functional specification review team was partially selected from the membership of the concept review team. Those retained were experts in their fields and had schedules that permitted participation. Other new members were added to strengthen representation from functional areas that were greatly affected by configuration management tasks. This review demanded a more detailed understanding of data associated with daily tasks and therefore required representation from the users with a working knowledge of the function.

The result of this review was an 1100-page document that defined the complete set of tasks required to accomplish configuration management. From this list, tasks could be selected for detail design.

Since the scope defined by the functional requirements specification would require 3 to 4 years to develop, an initial phase was defined for detail design and construction. The data flow diagrams developed in the functional requirements specification played an important role in the first phase of detail design as well as subsequent phases. They provided the complete set of requirements used to prioritize later system releases.

The relatively small scope of the initial phase not only provided near-term system startup, but also improved user involvement and motivation by providing a goal requiring project level planning for implementation. The phased schedule is illustrated in Figure 2.

DETAIL DESIGN SPECIFICATION

The detail design specification defined the user interface to the application system, procedures for use of the system, overall software architecture, and database design. The contents of the detail design specification were as follows:

1. Introduction—The purpose and scope of the detail design specification document
2. System Description—An overview of the entire system's functional design
3. Subsystem Description—Data flow diagrams of all system functions, procedural flows and narratives, and online transaction descriptions
4. Changes to Existing Procedures—A step-by-step procedure describing manual and system operations
5. Test Plan—Test objectives and methodology
6. Training Plan
7. Appendix—Logical database design, manual forms, review team minutes, and glossary

Because of the detail required in the design specification, data flow diagrams were insufficient to represent the required information. The strength of the data flow diagram is its hierarchical illustration of a major problem. When a detail design is being made, procedural flows, control points, feedback loops, and organizational responsibilities become important. Therefore, procedural flows were chosen to show how the system would be used in day-to-day operations. However, as

---

The draft functional requirements specification defined additional levels of data flow diagrams below those of the concept definition, emphasizing areas for potential system application. Task descriptions were then developed to define further many of the processes in the lower levels of the data flow diagram. The task descriptions defined an operation in terms of the input and output data and a set of processing steps.

The functional specification review team was partially selected from the membership of the concept review team. Those retained were experts in their fields and had schedules that permitted participation. Other new members were added to strengthen representation from functional areas that were greatly affected by configuration management tasks. This review demanded a more detailed understanding of data associated with daily tasks and therefore required representation from the users with a working knowledge of the function.

The result of this review was an 1100-page document that defined the complete set of tasks required to accomplish configuration management. From this list, tasks could be selected for detail design.

Since the scope defined by the functional requirements specification would require 3 to 4 years to develop, an initial phase was defined for detail design and construction. The data flow diagrams developed in the functional requirements specification played an important role in the first phase of detail design as well as subsequent phases. They provided the complete set of requirements used to prioritize later system releases.

The relatively small scope of the initial phase not only provided near-term system startup, but also improved user involvement and motivation by providing a goal requiring project level planning for implementation. The phased schedule is illustrated in Figure 2.

DETAIL DESIGN SPECIFICATION

The detail design specification defined the user interface to the application system, procedures for use of the system, overall software architecture, and database design. The contents of the detail design specification were as follows:

1. Introduction—The purpose and scope of the detail design specification document
2. System Description—An overview of the entire system's functional design
3. Subsystem Description—Data flow diagrams of all system functions, procedural flows and narratives, and online transaction descriptions
4. Changes to Existing Procedures—A step-by-step procedure describing manual and system operations
5. Test Plan—Test objectives and methodology
6. Training Plan
7. Appendix—Logical database design, manual forms, review team minutes, and glossary

Because of the detail required in the design specification, data flow diagrams were insufficient to represent the required information. The strength of the data flow diagram is its hierarchical illustration of a major problem. When a detail design is being made, procedural flows, control points, feedback loops, and organizational responsibilities become important. Therefore, procedural flows were chosen to show how the system would be used in day-to-day operations. However, as

---

The draft functional requirements specification defined additional levels of data flow diagrams below those of the concept definition, emphasizing areas for potential system application. Task descriptions were then developed to define further many of the processes in the lower levels of the data flow diagram. The task descriptions defined an operation in terms of the input and output data and a set of processing steps.

The functional specification review team was partially selected from the membership of the concept review team. Those retained were experts in their fields and had schedules that permitted participation. Other new members were added to strengthen representation from functional areas that were greatly affected by configuration management tasks. This review demanded a more detailed understanding of data associated with daily tasks and therefore required representation from the users with a working knowledge of the function.

The result of this review was an 1100-page document that defined the complete set of tasks required to accomplish configuration management. From this list, tasks could be selected for detail design.

Since the scope defined by the functional requirements specification would require 3 to 4 years to develop, an initial phase was defined for detail design and construction. The data flow diagrams developed in the functional requirements specification played an important role in the first phase of detail design as well as subsequent phases. They provided the complete set of requirements used to prioritize later system releases.

The relatively small scope of the initial phase not only provided near-term system startup, but also improved user involvement and motivation by providing a goal requiring project level planning for implementation. The phased schedule is illustrated in Figure 2.

DETAIL DESIGN SPECIFICATION

The detail design specification defined the user interface to the application system, procedures for use of the system, overall software architecture, and database design. The contents of the detail design specification were as follows:

1. Introduction—The purpose and scope of the detail design specification document
2. System Description—An overview of the entire system's functional design
3. Subsystem Description—Data flow diagrams of all system functions, procedural flows and narratives, and online transaction descriptions
4. Changes to Existing Procedures—A step-by-step procedure describing manual and system operations
5. Test Plan—Test objectives and methodology
6. Training Plan
7. Appendix—Logical database design, manual forms, review team minutes, and glossary

Because of the detail required in the design specification, data flow diagrams were insufficient to represent the required information. The strength of the data flow diagram is its hierarchical illustration of a major problem. When a detail design is being made, procedural flows, control points, feedback loops, and organizational responsibilities become important. Therefore, procedural flows were chosen to show how the system would be used in day-to-day operations. However, as
each phase of detail design was initiated, the data flow dia-
magrams were consulted for information on interfaces required
within the configuration management system as well as for
other systems.

User Review

The review team consisted of 12 members of the user com-
munity representing five disciplines. Some of these members
had been on the functional requirements team, but new team
members were needed to challenge the work of the earlier
team.

As is typical in system design reviews, the format and usage
of each online transaction was described and discussed in
detail. User procedures, including manual and system oper-
ations, were also presented during the eight-session review.
Since the application was to be developed using a database
management system, the database design was reviewed with
users. Figure 3 illustrates how users were involved in such a
system-oriented topic as database design. The database de-
sign was presented by using colored dots relating each data
item on every screen to its location in the database. The users
were then asked to evaluate the database design to insure that
1. All items of importance (entities) were identified.
2. All attributes of each entity were identified.
3. All relationships (one-to-one, one-to-many, many-to-
many) between entities were properly identified.
4. Potential future requirements would fit within the over-
all structure.

During the database review several attributes were found to
be associated with the wrong entities, but the overall structure
remained unchanged. The users did, however, express com-
fort at having a better understanding of this hidden aspect of
system design.

Design Verification

After the design review there was doubt that the design
had been reviewed thoroughly enough to be certain it would
function in an operational environment. The concern was due
to the significant changes required in manual procedures,
coupled with the users’ lack of experience with the use of
computer systems. An additional test was added to the design
process to validate the approved design further.

The design verification test manually simulated the system
operation. The test goal was to determine whether all neces-
sary data had been captured and the proposed manual pro-
cedures would work. Six areas were chosen to participate;
some representatives were not from the detail design team. By
using paper versions of the proposed online transactions,
users were asked to operate the system in parallel with their
current procedures. Although execution of this test was ini-
tially sporadic, the systems analyst was able to provide suf-
cient motivation to obtain a complete set of test results. The
results are shown in Table I.

Development Plan

Before the system was constructed, the complete schedule
was developed for the construction and user test phases of
system development. The format of the schedule was such
that the users would understand not only what was going on
within the development organization, but also their responsi-
bilities during the development cycle. At this time specific
users were identified to assist in the construction and test

<table>
<thead>
<tr>
<th>TABLE I—Design verification test results</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMBER IDENTIFIED</td>
</tr>
<tr>
<td>DESIGN PROBLEM DESIGN</td>
</tr>
<tr>
<td>IMPROVEMENT PROCEDURAL PROBLEM</td>
</tr>
<tr>
<td>PROCEDURAL IMPROVEMENT</td>
</tr>
<tr>
<td>TRAINING PROBLEM</td>
</tr>
<tr>
<td>TRAINING IMPROVEMENT</td>
</tr>
<tr>
<td>TOTAL</td>
</tr>
</tbody>
</table>
phases, and the first group was identified for initial production startup of the system. Joint commitments were made by developer and user—the developer to meet the specification and the schedule and the user to implement the system on a project. User responsibility for the system had shifted from the user community at large to the specific users’ own operating organization.

SYSTEM DEVELOPMENT

The construction phase consisted of six major milestones for each transaction being developed. First, a preliminary design review was conducted between the analyst and the assigned programmer to insure that details of the design specification and the overall systems architecture were understood. The next step was the development of the user documentation for the transaction by the programmer.

A review of user documentation was then conducted between the developer and user identified in the development plan. This was a key review to insure that the programmer understood the design specification as well as any implications not detailed in the design specification. It also gave the user a final chance to make modifications to the design prior to coding, and it established a working rapport between the programmer and the user. (Though user documentation is seldom written before coding, the method is widely recognized as a good one. In addition to the benefits mentioned above, it maintains user involvement at a point in the development cycle that is typically void of any involvement. In this case the maintenance of involvement was extremely beneficial in preventing the common attitude among users that after the specification is complete, user participation stops.)

The approved draft of the user documentation was sent to the technical writing staff for editing while the programmer continued with the construction phase. The next step was program design and pseudocoding, which concluded with a program design review. After source coding was completed, a code walkthrough was conducted with other members of the programming staff. Upon approval by the lead programmer the assigned programmer completed testing of the program and preparation of the test package for the user unit test. This test package consisted of the completed user documentation and transaction test instructions.

User Test

The test package was used by the user identified in the development plan to help in understanding the transaction and the purpose of the unit test. This test averaged a cycle time of less than 2 weeks for each transaction.

Once all programs had passed the unit test, the users conducted a system integration test to insure proper communication of data among all the transactions within the system. This test required about 2 weeks to complete.

The final test was the parallel production test, which, unlike prior tests, used real data in the operational environment. Before beginning the parallel production test all users participated in a training class, which allowed a prototype test of the training material. The parallel production test lasted for about 6 weeks.

During the entire testing cycle, weekly meetings were held in the user’s area to review test results. All comments were documented for future disposition. User reaction to the testing was favorable. Users felt they could affect the design of the production system and therefore were motivated to complete the testing. The success of these efforts is presented in Table II, which displays the program changes identified during design as compared to those identified during construction and test. The results show that the majority of changes were identified during design, indicating a successful communication of the design between the developers and the users.

Implementation and Fanout

The development plan was executed on schedule, and the system was put into production with no major problems. After a month of use on the first project area, the system was installed in a second area. Several minor problems arose during the first month of use on the second project. However, users from the first project volunteered to meet with users from the second project to assist them in several procedural issues. This cross-communication between users was helpful in achieving user acceptance of the new system and procedures.

Several months into system operation a major procedural issue was identified by one of the functional user groups indirectly affected by the system. The issue should have been resolved during the review of the detail design specification. It was discovered that the user who represented the functional area affected had attended less than half of the design specification review meetings. Although this issue was resolved without any major impact, the occurrence did point out that particular attention must be paid to insuring that the users accept the responsibility that goes along with the authority they have as team members representing their respective user communities.

RESULTS

During the concept and functional requirements specification activities, data flow diagrams were a key in providing a struc-
ture to maximize effective use of the users' time. Beginning with the detail design specification, use of procedural flow and control flow diagrams replaced the data flow diagrams. Following the user review of the design specification, several ad hoc techniques were used to maintain a constant level of user involvement and interest during the construction and test cycle. These techniques included design verification test, development planning, user documentation walkthrough, and unit and parallel production test.

The techniques communicated to the users progressively more detailed perspectives of the system design, allowing them to visualize and critique the system. The frequency of these milestones permitted the users to monitor the progress of the development activity. As the design evolved, the users saw the incorporation of their ideas into the system design and felt more sure that the system would succeed in solving the problems in their environment.

The 50 to 60 users involved in the various development activities had a major stake in the successful implementation of the system. As shown in Figure 4, the user contribution was 30% of the total effort and was spread across all phases of development. Since their involvement was voluntary (and in addition to their regular job), the amount of their contributions illustrates their acceptance of responsibility for their role in the development activity. As the system design evolved, this responsibility grew—to such a degree that the users felt that the system was theirs by the time of implementation.

CONCLUSION

Tools and techniques for systems development should be selected to support frequent communication between the developers and the users. It is not the particular tools that are important, but the selection of a set of tools that improve the users' understanding of the design as it evolves.

REFERENCES