

Starting a Requirements Management Initiative

*Mike Young, Don Moreaux, and Kris Hartung
Hewlett-Packard Company
11311 Chinden Blvd., Boise, Idaho 83714-0021
Mike_Young@boi.hp.com
Don_Moreaux@boi.hp.com*

Abstract

Requirements become the basis for all of a project's subsequent management, engineering, and assurance activities. Consequently, how effectively requirements are managed can have a dramatic influence on project resources and schedules, as well as product quality.

This paper describes one HP Division's experiences in starting an initiative that significantly changed the way in which its managers, engineers, and marketing specialists understand and implement *product requirements management*. The experiences detail how, by maturing our requirements management practices, our organization was able to address the following issues: 1) information redundancy and synchronization, 2) inconsistent documentation format and content, 3) inefficient requirements elicitation, communication, and review procedures, and 4) lack of customer focus. The paper also offers recommendations for those organizations that plan to start a similar initiative. Finally, it discusses plans to improve our requirements practices on future projects.

Introduction

Hewlett-Packard's Personal LaserJet Division produces a family of software, printer, and multi-functional products that are designed for home office, small office, and personal use environments. In the past, we have maintained small teams and simple products, and have been able to produce highly reliable solutions with a simplistic and uncontrolled documentation and requirements management process. We recently changed our product development strategy by focusing on component development and by re-organizing into a matrix organization to create a more common solution and lessen duplicate work. This has created parallel development of components and a fairly complex reporting/working structure with multiple communications paths.

To address this increased complexity of our organization and products, we determined that it was necessary to invest significantly in creating a solid set of product requirements and some means of effectively managing them. The purpose of this paper is to share what we have learned as an organization during the evolution of our product requirements management process. We describe the difficulties that we encountered with our previous approach and how we solved these problems with our implementation of a more efficient and sophisticated tool-based approach. Finally, we outline some additional benefits that we realized, which we had not considered prior to the initiative, and discuss our plans for further improvement in developing the next generation of products.

The Process Prior to Starting the Initiative

We have always launched new product development by assigning a “program manager” to investigate the feasibility of a new product idea. Although extensive research of customer needs and market opportunities did occur prior to the initiative, the information was typically only captured in a high-level data sheet for the product. We made a large jump from the data sheet to low-level engineering design of each subsystem – first hardware, then firmware (embedded software), and lastly software began fitting their designs into the already-defined work of the other teams. The system specification occurred piecemeal throughout the development cycle of a product. For the most part, our requirements management process consisted of a data sheet, sporadically created and updated documents, and a process of requirement elicitation that we had reduced to informal management-driven communication between marketing and our research and development lab.

The major problem with this ad hoc requirements process was that it lacked *well-defined* and *documented* requirements. The end result was that we addressed many issues too late in the process to make changes, products tended to be very technology focused, and each subsystem became an “add-on” rather than an integral piece of a fully architected system designed to meet specific customer needs. Schedules between teams were difficult to coordinate, and we integrated the complete system for the first time at the very end of the development phase. We were still able to produce high-quality products that met customer needs, but doing so required frequent heroics and long test cycles.

The Initiative

Defining the Problem

One practice that is worse than leaving the development process alone is to try to roll out a solution that does not solve a known problem. When we performed defect “root-cause” analyses on three separate project teams, they all pointed to the lack of requirements as the top cause of defects. Resolving these defects required changes in product design or substantial rework of algorithms or test code. This and the overall frustration level converged and made us realize that we needed a well-defined document strategy and requirements management process.

The initiative we undertook attempted to solve the following key problems that we saw in our past product development. We felt this would greatly increase our effectiveness as an organization and our ability to deliver quality products on time.

- 1. Information Redundancy and Synchronization.** Product documentation included information that each author owned along with information that they believed affected their share of the product design. Consequently, the same information was recorded in many documents because every author was documenting part of every other author's work. With numerous places for the same information, documents became out-of-sync quickly, and no one knew the location of accurate data.
- 2. Inconsistent Documentation Format and Content.** Because authors often combined their own innovations with data from previous projects, everyone considered themselves the sole caretakers and distributors of that area of information. Reviews often became personal because documents were *personalized*. Document reviewers did not feel responsible for making sure that the information was correct because they thought that this was the author's job.

- 3. Inefficient Requirements Elicitation, Communication, and Review.** No one knew what documents were "official" for the program, where to obtain the latest version of each document, and whether the documents were up to date. In fact, documents were typically reviewed early in the project then not updated or put under change control until the product release. During product development and testing activities, word-of-mouth communication was the predominant mode of exchanging accurate information. This was typically a "pull" model as well, so that if other developers or testers did not know that they could ask for information, they did not find out about changes until integration or test execution. By relying on verbal communication, there was no definitive way of knowing whether we were really building what we initially set out to build. With a newly reorganized matrix organization that needed to do parallel development and work out many dependencies, verbal communication would not suffice. Many developers were conscientious about having documents reviewed up front, but the documents typically did not get updated until the product release and often lived somewhere in authors' local directories.
- 4. Lack of Customer Focus.** Because documents were always "technical descriptions," either of how the system worked or of how it was built, we did not seriously consider having a method for describing and documenting customers' needs and how they would use the product. Products were more driven by technology than by customer needs. There was ample validation by real users through usability and beta testing, but it was always too late to make serious changes when we discovered problems.

TBI's Caliber-RM Tool: Formulating the Process

As we began to develop a new generation of products, we started with a single requirements engineer who worked closely with multiple program managers to elicit and record preliminary product requirements. This initial working relationship proved crucial in getting program managers' buy-in and understanding of requirements management.

The initial deliverable in the above process was a single document based on a customized IEEE specification for a "Product Requirements Document." We collected requirements by gathering information from numerous sources—specifications and User's Manuals from previous products, requirements that other people had written down for other LaserJet products, and discussions with individuals. The main idea was to "get something in writing" because gathering and organizing information is often the hardest part of implementing a new process (it is easy to engage others to correct what is already there, but can be hard to acquire new information from them).

We made it a policy that all product specifications would be part of our product requirements document. Although this provided a single source of data and a consistent writing style, we very quickly found that we could never get or keep the entire document up-to-date or efficiently communicate the changes throughout the organization. This and the desire for a central archive of data compelled us to search for a tool that would help guide our activities and processes.

While we were searching for a requirements management tool, another LaserJet division had already started to manage their requirements by piloting a tool called Caliber-RM from Technology Builders, Inc. (TBI). After conducting a thorough analysis of requirements management tools earlier in the year, they chose Caliber-RM for its ease of use and its view of each requirement as a "database element" rather than a part of a document. This "database" approach promotes flexible reporting, shows traceability between requirements, and makes it easy to associate other fields with a requirement. We also liked the ease-of-use, the full change history for each requirement, the automatic e-mail notification to "responsible persons" when requirements change, and the fact that team members can provide feedback on requirements through a threaded discussion group feature accessible through a Web Access component.

The fields we used for each requirement are listed in Table 1. Having the same fields for each requirement and using pull-down choices where possible solved our "data inconsistency" problem. With the information entered into a central repository and split out by these fields, we ensured that the right information was captured and that it was highly visible and accessible to everyone. Through a reporting feature, the tool could also create various views of the data – by product, by status, by priority or ranking, or by filtering the data, such as generating a list of just the recorded open issues.

Table 1: Information tracked for each requirement

Field	Description/Values
Requirement Name	Short identifier used for summary reports
Owner	The individual who owns and keeps the requirement up to date
Status	The state of the requirement in the development cycle (Proposed, Approved, Scheduled, Implemented, Postponed)
Priority	Overall marketing value of the requirement (Must, High Want, Want) Note: this is influenced by the Product Ranking and Difficulty fields
Description	Free-form field; includes the user need and the general feature description
Device Requirements	Description of the implications to the embedded software and hardware
Host Software Reqts	Description of the implications to host software
Open Issues	Free-form list of questions or ideas that need to be addressed
Closed Issues	Capture of how and when each open issue was closed
Applicable Products	The products to which the requirement applies
Product A Ranking	The requirement's importance to Product A's customer
Product B Ranking	The requirement's importance to Product B's customer
Difficulty	The effort required to implement the features and satisfy the requirement (Unknown, Easy, Difficult, Very Difficult)
Settings	Default values; range of values for a requirement parameter
Management Issue	Checkbox to identify that the issues require manager action
Risk	The risk to the program schedule (Unknown, Low, Med, High)
Source	The origin of the requirement - a team, individual, or previous product

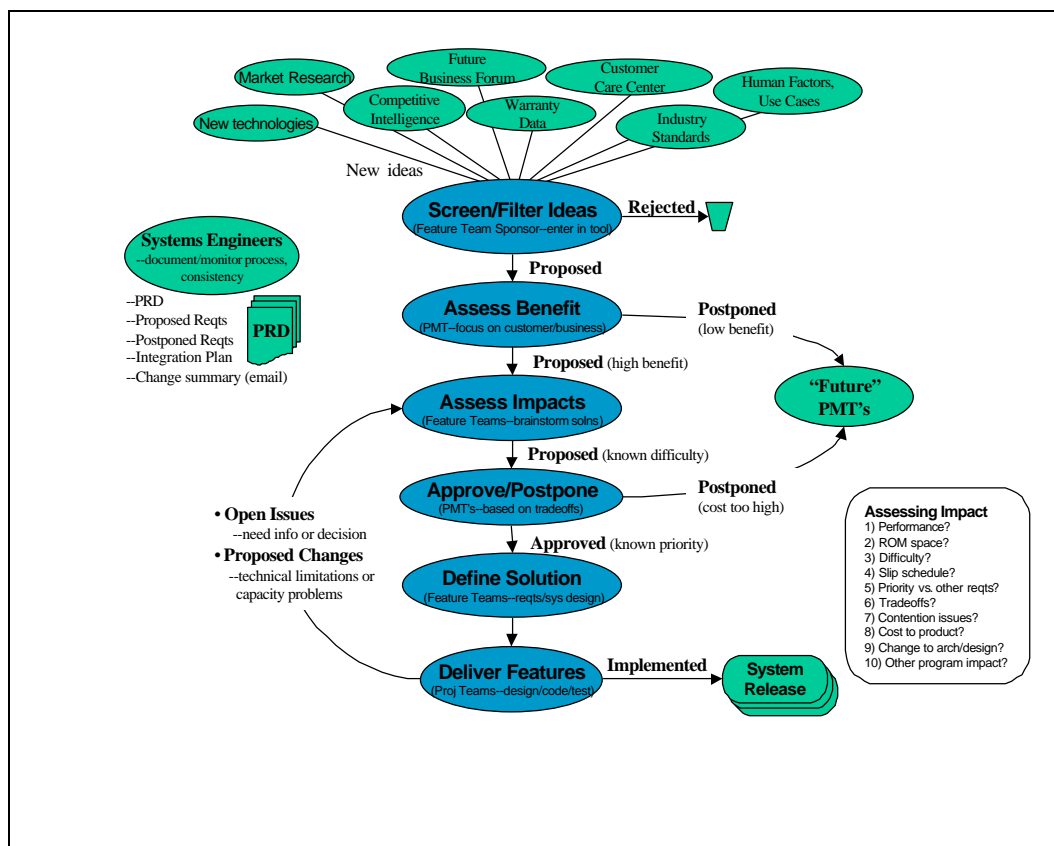
Because reports are generated automatically, the information is viewed as "belonging to the product" rather than "belonging to the individual owner." Writing requirements is much faster because requirement owners no longer have to focus on writing and formatting a long document, but on simply filling out the appropriate fields. Furthermore, by using the security aspects of Caliber-RM, our change management process has improved significantly by restricting the write access of each requirement to its owner.

Organizing around Communication: Rolling out the process

"Success or failure of a system depends on whether the relationships among its human components are as good as they can be -- even more so than the relationships among its technical components." Human engineer Cavett Roberts states that "85% of success in any field depends on knowing humans and human relationships" (1). Getting the right people to work closely together was the key to a successful rollout of our requirements management process. We started with a team of **requirements engineers** who provided oversight, support, and a consistent approach. We needed a screening/decision-making team who could assess customer benefits and determine whether to approve new requirements. Already existing **Program Management Teams (PMT's)** took on this role. Finally, we needed a team who could analyze all the incoming requirements, determining impacts, and devise design alternatives to meet the requirements. We formed cross-discipline **feature teams** to do this impact analysis and system design. They also allocated the requirements across each discipline. Figure 1 shows the process flow between these three teams to enable the requirements process. The process focuses mainly on the state of each requirement -- progressing from *proposed* to *approved* and from *scheduled* to *implemented*. It also includes the sources of most requirements. The key to the success of this process was making sure that

ownership existed at all levels. Reports were used to generate "to do lists" for each group to ensure that requirements progressed as necessary.

Figure 1: Requirements Management State Model, Roles and Responsibilities



Controlling Change

Currently, our product requirements are documented, controlled, and include project objectives, features, and performance characteristics. Commitments resulting from the requirements are negotiated with the affected groups and individuals. We make changes to controlled project requirements through the approved process in Figure 1.

For the formal requirements phase of the lifecycle, we always tried to err on the side of "keep it simple," so we did not focus on controlling change to the requirements. In fact, we typically did not communicate changes to the whole organization. Consequently, many of us were adding and changing requirements so quickly that it would have been overwhelming for anyone to have to take account of all requirements changes. Because all of the data was in a robust tool, we did capture a full change history of each requirement. The feature teams actually managed the whole process during the first phase, including brainstorming/collecting ideas, assessing benefit, and designing possible solutions.

After we transitioned out of the requirements phase by holding a large-scale review of all of the 100+ pages of requirements, we fully employed the process in Figure 1 for two main reasons:

1. *To make sure that any changes were fully communicated.*
2. *To protect the feature teams from turmoil in requirements.* Engineers tend to spend time finding solutions to any problem they are aware of rather than ensuring that customer needs are met first. Putting the Program Management Teams in charge of this "screening" process allowed

them to engage the engineering community on the feature teams on an as-needed basis to understand impact and possible designs.

We are working towards a weekly e-mail generated report from the tool that shows all changes to any requirements in the past week.

Impact

This section describes the details of the major impacts that our requirements management initiative had on our organization at the program level. Typically, our program managers coordinate the various functional areas, such as marketing, engineering, testing, manufacturing, quality, and support. Specifically, this section will focus on the positive influences that the initiative had on the program's subsequent management, engineering, and assurance activities. In some cases, the consequences were expected; in others, they ended up being favorable surprises.

The most surprising consequence occurred during the planned periodic review sessions of the Product Requirements Document (PRD). Membership during those review sessions was open to the entire program team. The main purpose during those reviews was to discuss new requirements for the product, as well as review and determine the priority and status (Proposed, Approved, Designed, Scheduled, Rejected, etc.) of existing requirements. Although there were long debates over rejecting or postponing high priority requirements, the decisions in most all cases ended up being final. This behavior was a big departure from the "it's in, it's out, it's in again" decision making that we were previously accustomed to when these decisions were made without the benefit of a program level review.

We were also surprised to discover that the way in which we categorized our requirements had two very interesting side effects. Our initial plan was to organize the requirements into groups, or requirement types, based upon product attributes. These types included such product attributes as Availability, Compatibility, Disposability, Functionality, Maintainability, Performance, Security and more. By doing so, we had hoped we could more reasonably ensure requirements coverage of the product.

However, the first consequence of this decision was that we found it very difficult to track requirements because some categories, such as Functionality, contained large numbers of requirements while categories such as Security and Safety were hardly used. The second consequence was that this categorization made it difficult to assign ownership for requirements, a necessary element of good requirements management.

Because the engineers on this program were already organized into cross-discipline engineering teams, called "Feature Teams", it became obvious to us that changing the requirement types to match product features solved two problems. First, by creating requirement types from our product feature set we were able to balance the distribution of the requirements. Second, because each of the Feature Teams controlled the design and implementation of its unique product feature, the issue of ownership was resolved.

Incorporating our Test Design Specifications into the requirements tool along with the product requirements provided additional benefits in the area of product assurance. The tool allowed us to automatically create and manage a Traceability Matrix (Figure 2), showing the relationship between the test requirements and product requirements.

Because the tool automates the ability to keep track of changes to either of these two pieces of information, test designers were immediately made aware of changes to requirements that might impact their testing. Additionally, test planning was now taking place much earlier in the development cycle than it ever had been on previous programs.

Figure 2
Traceability Matrix

	2.3 Custom Paper Size - PSWT4092	2.4 Number of Copies - PSWT4093	2.5 Paper Orientation - PSW	2.6 Pages per Sheet (N-up) - P
16 PostScript Level	↑			
17 Viewable Self-Te				
18 Self-Test Value C				
19 Maintain Country				
20 Change Fuser Te				
21 Booklet Printin			?	
22 Manual Booklet P				
23 Bitmap Watermar				↑
24 Text watermarks				↑
25 Font Symbol Sets				
26 4 Corlin Symbol				

Finally, we believe that we significantly reduced our documentation overhead by almost half. This was accomplished in a number of ways. Having a requirements management tool in place and providing easy network access provided a focal point for our organization. Everyone knew how to find the information, allowing them to reference the information instead of reproducing it in other documents. Report generation was automated, and we no longer spent hours formatting requirements documents in word processors.

Recommendations

Use a Tool To Help Manage Requirements

A wide selection of Requirements Management Tools is available on the market today. Although an organization can get by without such a tool, we observed that the benefits justified the cost of the tool we selected. The following capabilities will save substantial amounts of time in the course of your project's requirements management efforts:

- ◆ Track and Record History of Changes,
- ◆ Limit Write Access with Security Controls,
- ◆ Automatically Generate a Traceability Matrix,
- ◆ Create Customizable Requirements Forms, and
- ◆ Easily and Quickly Generate Customizable Reports

Have a Dedicated Requirements Engineering Resource

Eliciting and writing well-formed requirements is not a simple set of tasks. It involves a special skill set and a full time commitment to be done well. In our opinion, requirements engineers become proficient largely through experience. Requirements engineers also help manage changes to requirements when write access to the requirements is limited to them alone.

Use Guidelines as Aids to Writing and Reviewing Requirements

Currently, we write our requirements in natural language. Although requirements written in natural language are inherently prone to problems with ambiguity, inaccuracy, and inconsistency, the informality

of the language makes it a good candidate for specifying high-level, general requirements. However, this informality makes differences in the "formal" versus "popular" words and phrases that are difficult to use in precisely describing complex, dynamic systems. Even words with strict definitions, such as the IEEE-defined *error*, *fault*, or *failure*, are often used incorrectly.

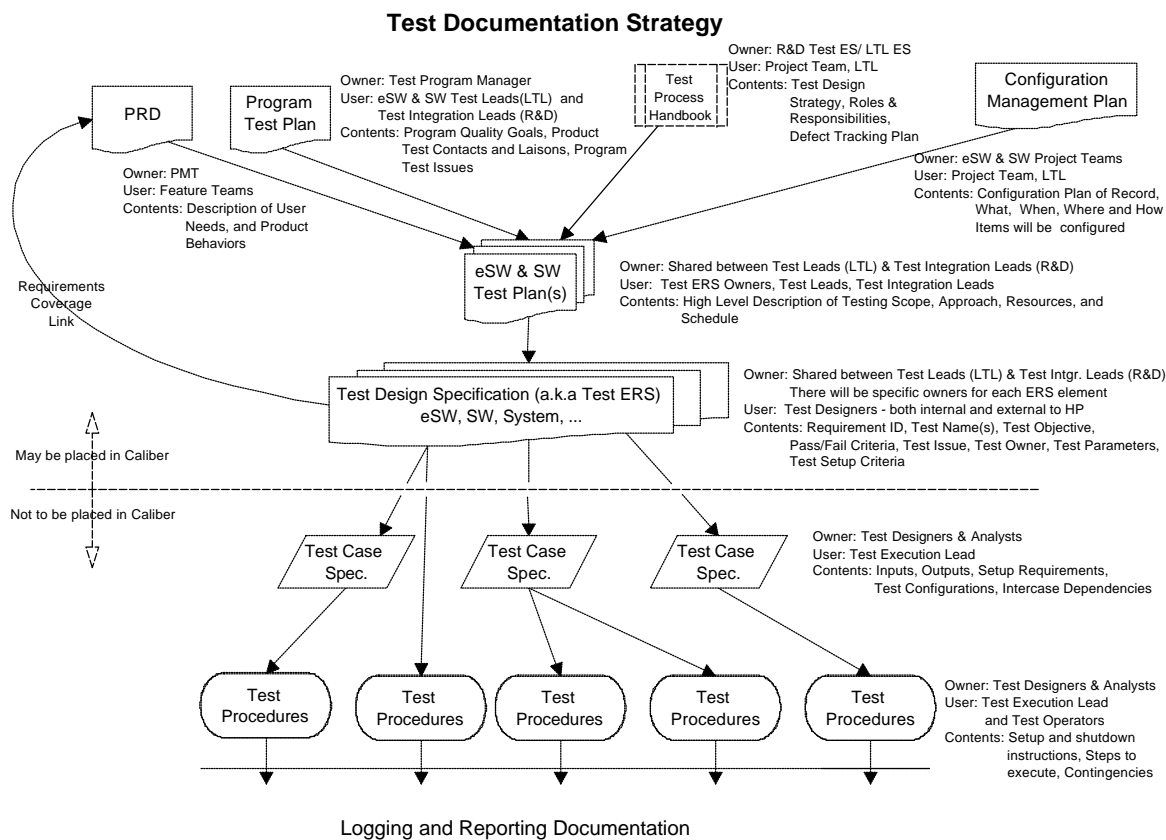
A set of guidelines for each requirements engineer has helped us avoid these problems [4]. These guidelines include the following:

- Give special attention to each word and phrase to avoid ambiguity and imprecision.
- Use the simplest language that is appropriate to the purpose of the requirement.
- Use the imperative voice, and avoid weak phrases such as "held to a minimum."
- Avoid generalities in place of number values (for example, "largest").
- Avoid words with relative meanings (for example, "normal" or "easy").

Decide on a Comprehensive Documentation Strategy

Product requirements eventually get "decomposed" into more detailed engineering-level requirements documents such as software requirements specifications, as well as test plans and procedures. Having a comprehensive documentation strategy, one that specifically identifies the hierarchy, ownership, and content, will greatly reduce the effort it will take to manage the suite of documents that will flow from the product requirements. An example of our test documentation strategy is given in Figure 3.

Figure 3
Test Documentation Example



Conclusions

For our next project, we plan to “start at the top” of the requirements management process by focusing on “use cases” for the product. In order to get an early start on requirements management for this initiative, we gathered information from every source that we could think of (documents from predecessor products, descriptions of competitive product offerings, and insights from sister divisions producing products). Rather than focus on the detailed specification first, we should make user needs more visible, and they should be driven by what we are trying to solve for the customer (based upon how customers are currently doing their jobs and how we can streamline that process). By describing the flow of events that we need to accomplish, we can drive the actual product requirements. By “driving product requirements,” we mean that user needs and use cases *become* the product requirements and shape the look and feel of product features rather than features being added just because they are a “cool technology” or a product of one engineer’s personal preferences. This is helping us change from being a technology-driven organization to one that is market-focussed and uses technology only to respond quickly to feedback from customers.

We need to establish owners early in the requirements management process. The key to the success of our initiative was assigning engineers to a particular feature area as the focal points for keeping the requirements up-to-date and coordinating all issues. This streamlined communication ensured that engineers were making progress. Proactively identifying the feature leads and their supporting teams at the beginning of new product development, rather than reactively waiting for a hole to appear in the process, would make a significant difference in the way we manage requirements.

For our next project, we will try to manage requirements early enough to include hardware requirements. The majority of the development organization consists of software development (host-based software and embedded software). This was the natural place to focus our development to gather requirement information that was readily available. For the next generation, we would like to have more influence on third-party companies delivering the hardware. This will involve specifying “quality goals” from a HP perspective rather than relying mostly on the specifications that these companies provide. Currently, we have a process of “improving hardware quality,” but we do not understand how to do this according to HP desires.

We want to add more sources of input to the requirements process. A number of established organizations in HP’s LaserJet organization could strongly influence requirements if we established the proper communication channels. We successfully included Human Factors and Market Research input; however, we could benefit by including information from the Customer Care Center (customer support), beta testing (from previous products), warranty information, and system test information. In addition, R&D is very tied to the “future product marketing” organization, which looks at future trends and conducts market research, but never obtains feedback from “current product marketing.” This feedback includes messages that are pushed with the media and major customer accounts, and major features or blocking issues that we should add/fix to be able to establish higher sales.

Next time we will use shared requirements. LaserJet devices share many requirements across the whole line of products, from low-end monochrome to high-end color (requirements like P JL, PCL6, localization needs, RFI testing, manufacturing test needs, etc.). Caliber-RM 2.0 (introduced mid 1999) added the ability to share requirement descriptions between projects, thus creating a “parent-child” relationship between the two. This eliminates duplicating information by re-entering and synchronizing data between projects.

Our project requirements should form the basis for estimating, planning, performing, and tracking the project’s activities throughout its life cycle. At our current speed of developing products, we recognize the value of effectively estimating and tracking activities. Using requirements as a source for this task is an area we are now seriously investigating.

References

1. J. Grady, *System Integration*, CRC Press, Boca Raton, FL, 1994.
2. J. Brackett, et. al., "Software Requirements Engineering," Software Engineering Institute Technical Course CE-SRE-01, Carnegie Mellon University, Pittsburgh, PA, 1992.
3. IEEE Standard 830-1994, *Recommended Practice for Software Requirements Specifications*, 1995.
4. W. Wilson, "Writing Effective Natural Language Requirements Specifications," *Crosstalk: The Journal of Defense Software Engineering*, February 1999.
5. M. Young, L. Simmons, Internal HP Requirements Documentation, 1998.
6. Technology Builders, Inc., *Caliber-RM Requirements Management User's Guide*, 1998.
7. Harold Kurstedt, "Human Components--The Greatest Challenge to System Success," *Proceedings of 13th Annual International Conference on Systems Engineering*, August 1999.