

Shaping Pachyderm 2.0 with User Requirements

Rachel Smith, The New Media Consortium, USA
Deborah Howes, The Metropolitan Museum of Art, USA
Wendy Shapiro, Case Western Reserve University, USA
Holly Witchey, The Cleveland Museum of Art, USA

Abstract

This paper describes a process for gathering user requirements used in the development of Pachyderm 2.0, an easy-to-use multimedia authoring tool with asset management capabilities. Examples of source documents, descriptions of procedures used, and specific results of user tests are included to illustrate the flow of the requirements-gathering and testing activities. The process can be easily adapted to projects of varying scope and budgets.

Keywords: user requirements; user testing; beta testing; authoring tool; Pachyderm

Introduction

The Pachyderm 2.0 Project is a partnership led by NMC: The New Media Consortium and the San Francisco Museum of Modern Art (SFMOMA), and funded by the Institute for Museum and Library Services (IMLS). The project brings software development teams and digital library experts from six NMC universities together with counterparts from five major museums to create a new, open source authoring environment for creators of web-based and multimedia learning experiences. The new tool will be based on Pachyderm, the authoring and publishing tool developed by SFMOMA to author its successful series, *Making Sense of Modern Art*.

The original tool was custom-developed by and for SFMOMA and is particularly suited to the needs of a modern art museum. Pachyderm has a work flow that reflects the team-oriented style used by SFMOMA to develop its innovative flash-based web features. The success of Pachyderm within the museum culture at SFMOMA led staffers to believe that the extension of the Pachyderm tool — in a revised format — to the museum and higher education community, could transform the way these key institutions present, share, and interpret content. To this end, SFMOMA staff applied for and won a National Leadership grant to create an open source version of Pachyderm that could be widely used and adapted by a variety of different audiences.

A tool with such a diverse user base presents a challenge in terms of design and development. Obviously, the requirements for Pachyderm 2.0 would differ greatly from the original specifications and requirements used by SFMOMA for the creation of the original tool. The team -- composed of museum professionals, software developers, pedagogy experts, educators, and others -- selected a user-centered approach for gathering requirements that would help to ensure the final product was as user-friendly as possible.

The process, which can be easily adapted to other projects, begins with collecting source documents and identifying gaps in the document library. Gaps are filled by activities such as scenario development and user observations. The sources are mined for requirements using a simple but thorough procedure. Requirements are collected, sorted, classified, prioritized, and

finally assembled into a requirements specification that guides development. This paper describes in detail how this process was applied to the development of Pachyderm 2.0.

Gathering Requirements

Source documents for Pachyderm 2.0 included existing historical documents such as the grant proposal, specifications for Pachyderm 1.0, and other pieces that had been written as the project was conceived. New data were gathered through user studies, persona- and scenario-writing exercises, surveys, and user observations; the documentation of these efforts joined the source library, and the bulk of the 300 or so requirements were derived from these. Requirements were captured from the assembled documents and recorded in a tracking database created for the purpose. For smaller projects, requirements could be captured in a spreadsheet or simply a text file instead.

All of the project teams (programming, metadata, user interface, and pedagogy) were involved in the requirements-gathering process. Each person brought a unique perspective to the process that helped to ensure a well-rounded set of requirements.

Envisioning the product with personae and scenarios

Despite the existence of Pachyderm 1.0, it was not clear at first exactly what Pachyderm 2.0 should look like. Very early in the project, the team used personae and scenarios to help create a unified vision of the product and to focus the development effort on making something that would appeal to end-users. One of the first steps taken by the requirements team was to identify primary user types for Pachyderm 2.0. Four primary types were identified: students, faculty, museum personnel, and museum visitors. Two secondary types were also identified: support staff and librarians. Other types of users exist but these are the major groups.

The "typical" Pachyderm user may have a lot of computer experience, or none at all. The expected age range is anywhere from a grade school student to a retiree. Note that there are two ways to "use" Pachyderm: authoring a module or viewing a completed module. In other words, there is no typical Pachyderm user -- it can be almost anyone, at anytime, doing anything. The development team envisions authoring not as belonging only to faculty or museum staff, but potentially to students and visitors as well.

Having identified these user types, the team developed several descriptions of specific but imaginary people (personae) for each group. Each persona includes the following elements:

- Biographical information about the imaginary person (name, age, occupation)
- Brief sketch of a typical day or other block of time
- Personality traits that affect how the person does work related to Pachyderm
- Information about his/her level of experience with technology

The personae were then used to focus scenarios, or descriptions of activities that take place while someone is using a product or service (in this case, Pachyderm 2.0). Keeping specific, familiar personae in mind while writing allowed the team to consider details that might otherwise have gone undiscovered. The background and experiences of each persona influence the way each scenario activity plays out, revealing new requirements for the end product.

Scenarios can be as brief as one sentence or as long as several paragraphs. Each scenario should include the following elements:

- Prerequisites (what has happened prior to this particular scenario?)
- One or more personae (who is involved?)
- Activities (what is happening?)
- Results (what is the ending-state of the scenario?)

Thinking through the scenarios helped the project team to think about how actual people will really use Pachyderm 2.0. At the scenario-writing stage, it is more important to focus on what the experience will actually be like from the point of view of the people in the scenario, rather than trying to pin down technical details of how the product will work. This can be difficult for developers, who are very comfortable thinking in terms of technical details; but it is essential to look at the product from the perspective of end users, especially during the early stages of requirements gathering.

Personae and scenarios for Pachyderm 2.0

Sample personae developed during the requirements phase are included below for each of the four major user groups. These personae are fictitious and bear no intentional resemblance to any individual person.

Student: Liam C., Freshman

Liam is a freshman at a small liberal arts college. He has not decided on a major yet, though he loves music and is an accomplished pianist. Liam is visually impaired and cannot read normal text on a computer screen. He can distinguish large shapes and can read if the type is very large. He learns easily through auditory means and has a roommate who will describe diagrams on the white board to him when they are in the same class.

Faculty: Chris F., Architecture Professor – Early Adopter

Chris is a professor of architecture and an early adopter of technology. She has taught for seven years and is up for tenure at the end of the year. She is always looking for better ways to teach and often uses her weekends to go online to see what new learning objects are available. She often uses MERLOT as a resource. She wants to pull together a strong syllabus this year, using a variety of media, and really wants to author her own presentation available online for others to use. (Giving back to MERLOT.)

Museum Personnel: Charles H., New Media Director

Charles is responsible for new media and works with curators who do not want to be Pachyderm authors but want to have him make projects for them. He needs to interpret the content delivered by the curator and convert it into something that will be easily understood and appreciated by the visitors. His technical skill level is high.

Museum Personnel: Jen T., Docent Trainer

Jen presents information about the museum's exhibits to her docents prior to the exhibitions. She is familiar with the museum's kiosks (from an end-user perspective) but does not use computers on a regular basis otherwise.

Museum Visitor: Carl M., Retiree

Carl is a sixty-eight year-old visitor who is retired and looking for ways to use his time that stimulate and interest him. He has an Internet account at the local library and does some limited e-shopping.

Museum Visitor: Marjorie R., Stay-at-home Mom

Marjorie drops her kids off in the art classes and likes to do something in the Museum while she waits for the classes to be finished. She believes in the arts and wants to have more information about the permanent collection. She has limited technical skills and does not use a computer at home.

Sample scenarios

The following scenarios are actual examples from the Pachyderm Scenarios Document.

1. *Creating a Module Using Placeholder Assets*

Persona: Charles H.

Prerequisites: The museum for which he works is hosting a major international exhibition on the Dukes of Burgundy; can Charles organize all his assets and use them to create a stunning presentation using Pachyderm?

The museum Charles works for is hosting a major international exhibition 11 months from now. Charles is beginning to gather assets—photographs, QTVR 360s of sculpture, interviews with curators—but he knows that much of the actual physical assets of the presentation will not be available until just prior to the exhibition—when the funding will be made available for his team to go over to France and do blitzkrieg photography of the objects in-situ and the landscape.

Charles has physically story-boarded the entire Pachyderm experience—but needs to actually build the system on the promise of photographs and videos that may or may not be available at the 11th hour. To this end, he has actually crafted two different story-boards, one with assets he is sure she will be able to get in some form or other and a second with the images he hopes to be able to get.

Even though he may not get the assets until the last moment, the curator he is working with is absolutely certain about the text he wants associated with assets and with how he wants the experience structured. Thus even though Charles cannot get the assets until the last moment, the curator wants to see a functional working model and sign-off on the text prior to his extended vacation in the Bahamas. Using the authoring tool, Charles builds both structures, assigning place-holder images—clearly indicated as such in the structure of the tool (for example sample.jpg, sample.gif, sample.mov, sample.ani). The resulting presentation includes all the appropriate text and layout, but the spaces for media have blank grey placeholders. When Charles gets the digital images later, it is easy for him to place them in the correct spots.

2. *Returning to a Saved Work-in-progress*

Persona: Chris F.

Prerequisite: Chris started work on her Winter quarter module in late summer, but hasn't worked on it all Fall; it is now two weeks from the start of the Winter quarter.

Chris can't remember the URL to the Pachyderm system, so she goes to her university home page and types in a Pachyderm search. It takes her to the Pachyderm site at her school, where she is asked to log in. She logs in, and she sees all of the departmental, shared modules as well as the module she started over the summer.

She clicks "edit" under the name of her module. She gets to the module overview screen, where she sees a storyboard view of her work in progress. She decides right away that she's ordered things incorrectly, so she drags two of the segments around to reorder them. She then decides to review the section on Temples. She enters it, and sees her side-by-side image comparison screen. She remembers that she hasn't finished adding her text commentary, so she pulls out her lecture notes from last year, and starts paraphrasing as she types in the commentary field.

Conceptualizing changes with user observations

The Pachyderm 2.0 team was interested in testing both the usability of a finished learning module, such as those found by visitors to the SFMOMA web site or galleries, as well as the usability of the production environment previously experienced only by SFMOMA staff (who created the system). Since 2.0 will be an open source, widely-distributed product it was important for the team to test both sides of the 1.0 system outside of the SFMOMA environment. As there was little time or money to hire outside evaluators to conduct the studies, team members volunteered to conduct observations and surveys in their own institutions, and in this way were able to study a wide spectrum of users and use cases for very low cost.

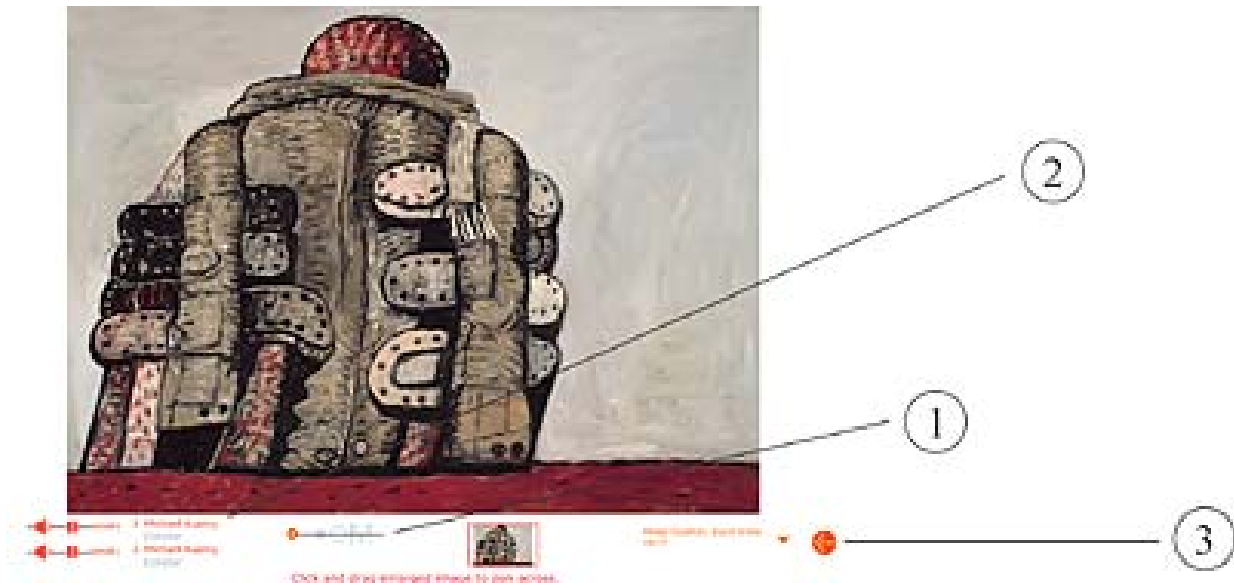
One of the great advantages of developing a 2.0 product is that there already exists a 1.0 from which to gather important information, and in the case of Pachyderm, SFMOMA had already produced many learning objects using the 1.0 platform (see www.SFMOMA.org). The Educational Media Department at the Metropolitan Museum of Art ("the Met") ran an assessment on the usability of the module created by SFMOMA for their recent Philip Guston exhibition, primarily because that same exhibition was on view at the Met at the time and recruiting museum visitors for the study would be facilitated by visitors' interest in learning more about this artist.

The Met study included Museum staff and visitors of all ages with varying degrees of technical skill and familiarity with the Web. The Met evaluation staff worked in pairs: one evaluator acted as facilitator, asking each user or group of users to simply go through the feature as if they had come upon it on a Web site or at kiosk in a museum. As in standard usability test procedure, users were requested to "talk out loud" and explain what they were doing as they were doing it. The facilitator-evaluator continually asked the user why certain actions were taken in order to clarify the reason for decisions made by the user. Users were forthcoming with feedback on what they liked and did not like, and whether or not they found the features on each screen to be effective and clear.

The other evaluator in the role of scribe wrote down all comments and actions taken by the user onto a form that had pre-printed thumbnails of all of the Pachyderm templates. Using this form the evaluator scribe could quickly associate observations with the correct template screen as the user navigated the site. At the conclusion of each session, both evaluators went over the forms and filled in any other details that were not captured during the usability session.

After 20 usability sessions of approximately 45 minutes each, the observation component was concluded. The evaluators concurred that users were repeating mistakes, observations, and suggestions that previous users had already made and the evaluation team was recording very little new information. As observation information had been recorded according to template type (using the prepared form) the data was already sorted in a useful way and therefore very easy to analyze and summarize into succinct descriptions of usability findings

for each of the 12 templates. The figure below shows an example of one of the screens, followed by the description of the results of the observations.



Pachyderm 1.0 Zoom Screen: one of the templates examined in the user observations.

This screen presented several problems. Even with the directions, many of our users did not understand that there was a zoom feature on this template (1). However, once they did figure out how to work the zoom, they liked the results. Many found the manner in which the audio content was indicated and described to be confusing. They questioned why there were two sets of controls, and felt that more descriptive labels were needed (2). Also, many of our users were confused by the location of the back button at the bottom of the page, as opposed to the usual location on other screens at the top right (3).

The report also included comments pertaining to overall look and feel and suggestions for improvements. For example, users complained about the small size of the window (dictated by an earlier incarnation of Flash), and the frequent use of the decorative orange frame as a place for important directions ("drag the circles together...") seemed too subtle for many visitors. Some navigational buttons were difficult to find and understand as their placement was inconsistent and their shapes were similar to buttons used for other kinds of information. Users asked for a wide range of new features such as breadcrumbs (a tracking system common to web navigation by which visitors see a list of the pages traveled and click on a page name to return there), an index, and a print-friendly screen option. It was interesting to note that although users were tested at a kiosk in a museum, they expected the Pachyderm templates to operate as Web pages, with, for example, a back button in the same place as a web browser and text characteristics that allowed for cut, paste, and easy searching.

Deriving requirements from scenarios and user observations

The process of transforming a group of documents into usable development requirements is time-consuming but not difficult. Continuing with the examples presented, this paper discusses how requirements were mined from the scenarios and user observation records.

From scenarios to requirements

By reading between the lines of the scenarios, the team can start to compile a list of *assumptions* about the product, user and system *requirements* for the product, and *issues* that must be resolved.

Assumptions are basic facts or limitations that the development team must work with. For example, some projects assume that all access will be via a web browser. Issues are unresolved questions; these questions will need to be answered, either by finding a solution or by deciding not to include a given feature in the current version of the product. Both assumptions and issues should be noted in the requirements document.

Requirements are the basic functions that a system needs to be able to carry out in order to perform the task(s) for which it was designed. The meat of a requirements document is, of course, the list of requirements. *User requirements* are functions that someone using the system will see or use. *System requirements* (also called technical requirements) are behind-the-scenes, technical functions that the system must be capable of doing in order to support the user requirements.

The following questions are helpful when pulling requirements out of scenarios:

- What is the user trying to do?
- What part of that task is facilitated by the system or product?
- What part of that task is independent of the system or product?
- What has to happen "behind the scenes" while the user does this task?
- What does the user see on the screen while s/he is working on the task?

The answers to these questions will lead to statements of user and system requirements. For example, applying the questions to the second scenario above:

What is the user trying to do? The user wants to open a module she began some time ago and pick up where she left off.

What part of that task is facilitated by the system or product? The system stores her partially-completed module and gives her access to it. (Implied requirements: there must be a way to save work in progress; there must be a way to find and return to saved work.)

What part of the task is independent of the system or product? Finding the URL or link to the Pachyderm system is not part of Pachyderm itself and implies no system-related requirements. Often, scenarios will include "flavor text" that helps to set the stage but that has no direct bearing on requirements.

What has to happen "behind the scenes" while the user does this task? The system has to "know" who Chris is and which modules are hers or accessible by her. The system has to allow her to rearrange her screens somehow. The system has to be able to accept text that she types into a field. (Implied requirements: there must be persistent user accounts with logins; users must be associated with presentations they have created and/or have access to; screens can be reordered in the authoring system; the authoring tool allows users to type into text fields.)

What does the user see on the screen while s/he is working on the task? Chris sees a list of modules she can edit. When she picks one she sees all the screens in her module; later she looks closely at one screen in particular. (Implied requirements: the system must list user-

editable modules; the user must be able to select one module to edit it; the system must display an overview of a module showing all screens; the user must be able to select one screen to edit; the system must display an editable version of the selected screen.)

From user observations to requirements

It was very easy for the Pachyderm team to derive requirements from the observation report written by the Met staff as it contained the range of specific data (rationale, stakeholders, etc.) needed to complete the requirement listing. In a day-long group meeting, team members took portions of the report, translated every observation into a requirement and gave each a category and a priority. For the most part correcting navigational flaws were given high priority ("essential") while suggestions for new features were given a lower priority ("desirable") especially if it was perceived by the group to be a difficult task for the Pachyderm platform to support.

Working as a group allowed questions about categorization and programming implications to be discussed by a wide range of team talents. For example, the users' request to have cut and paste functionality within a module seemed modest to an educator but the programmers who are more familiar with the limitations of the Flash platform and anticipated version enhancements had another opinion. During the requirement building process team members frequently sought input from others so that each entry could be as complete as possible. Having complete information about each requirement greatly facilitated the next step of refining the list and establishing an overall priority.

Sample requirements

The following requirements are actual samples from the Pachyderm 2.0 Requirements Specification.

3.1.1.1 Persistent user accounts (#29)

The system must support persistent user accounts, allowing users to log in (and out as needed) to access data tied to their account.

Rationale: Users will have basic data that should be "remembered" by the system so users don't have to look it up or enter it by hand each time they use Pachyderm.

Notes: This refers to users of the Authoring Tool (not to users of completed modules).

Volatility: stable

Clarity: refine (We need a good definition of what data would be tied to their accounts.)

Recorded by: Rachel Smith

System Components: Authoring Tool

Stakeholders

Developers: Programmers, User Interface Designers

End Users: Authors (Education), Authors (Museum)

Official Priority: essential

Desired Priority: useful

3.1.1.16 Return to a saved work-in-progress (#195)

Authors can come back later to work previously begun.

Rationale: No one works in one sitting.

Notes: (no notes)

Volatility: stable

Clarity: clear

Recorded by: Jared Bendis and Claire Stewart

System Components: Authoring Tool

Stakeholders

Developers: User Interface Designers, Programmers

End Users: Authors (Education), Project Managers, System Administrators, Authors (Museum), Production Staff

Official Priority: essential

Desired Priority: essential

Dealing with Requirements

Once the requirements had been pulled from the sources and entered into the database, they were carefully refined, sorted according to which team would be dealing with each requirement, and prioritized. This process took several weeks to complete, and the resulting document (nearly 100 pages) served as the "road map" during development. Naturally, requirements evolve during development, and a change process was in place to handle this. The requirements document also serves as a checklist during post-development system testing.

Writing, sorting, and prioritizing requirements

A requirement begins as a statement of something the system must supply or be capable of doing: *The system must support persistent user accounts.* As the requirement is examined, the original statement may be refined to make it more specific. It is also useful to collect other data along with the requirement, such as a rationale describing why the requirement is needed, the name of the person who recorded the requirement, the users who will be most affected by the requirement, and so on.

Most of the work of capturing requirements for Pachyderm 2.0 occurred at a single face-to-face meeting. Using wireless connectivity and the online requirements tracking database, the team divided the source documents and worked in small groups to pull as many requirements as possible from the sources. Each requirement was entered into the database, along with the rationale, the name of the source, the names of the people recording it, the desired priority of the requirement, and other information. The team spent two days doing nothing but capturing requirements. At the end of that time, over 200 requirements had been recorded.

The next step was to sort the requirements according to the part of the system to which they pertained. This was done at a distance over a period of several days using the database. Each team was responsible for one part of the system; the teams then went through their requirements to classify them and assign official priorities. Requirements assigned an "essential" priority are to be completed in Phase One (by Oct. 2005). A draft requirements document was compiled, which was reviewed by all the teams and discussed in detail at the 2004 Spring Meeting (April 2004). This document may be downloaded at www.nmc.org/pachyderm/docs/PachySpec1.5.pdf.

Validating requirements

Once the teams agreed on the draft document, it was submitted to volunteer readers from each of the core user groups. The readers who were selected had varying degrees of comfort and experience with technology. They were asked to review some of the requirements (not all; many are very technical in nature) and comment on the features that would be included. Specifically, they were asked whether the included features were desirable as written;

whether any important features seemed to be missing; and whether the priorities assigned to features matched with their needs (i.e., were there any features the users thought were critical that had been prioritized as lower than "essential"?). Results of the validation were positive; the feature list in the requirements specification matched users' expectations and desires.

Requirements evolve

All good program teams recognize that requirements for a project are seldom written in stone. As soon as actual development of a tool begins, the requirements themselves begin to change. The challenge for the Pachyderm team, as for any team, was to keep a handle on the evolving requirements and to make sure that changes in requirements did not, in any significant way, change the goal of the actual project: the creation of a flexible tool that could be easily used to create content-rich presentations with a simple but beautiful set of highly intuitive human interface templates.

One aspect of the requirements that has become increasingly apparent as the beta test has progressed is the need for a common vocabulary of both literary and visual terms to describe both the templates and the actions encompassed by the templates. SFMOMA had developed an internal vocabulary and work-flow which enabled them to work with the original Pachyderm, but as the partners for Pachyderm 2.0 came from a variety of museums and institutions of higher learning, use of SFMOMA's proprietary phrases like "the artist-in-context screen" and the "phone-dial" caused some confusion. This ultimately led to a discussion to determine a more commonly agreed-upon set of names for the individual templates in the major template set that reflected the broad scope of Pachyderm 2.0. This is just one example of an initiative that was not a requirement which came out of the original scenarios but evolved as the team needed to find a common language to discuss this new tool that was being created.

Other requirements evolved as the development team discovered limitations to, or possibilities created by, the technology used to create the system. Further testing of the new interface design revealed requirements that were not uncovered in the first round, and these went through a screening process before they were added to the requirement listing. Protecting the integrity of the requirements document without being inflexible is a delicate balance; having a specific change process in place, ready for issues that might come up, helped a great deal.

The change process used for Pachyderm 2.0 was relatively simple. The proposer of the new or changed requirement wrote up a brief description of the change, a rationale for making the change, and a list of affected end-users. The requirement was classified and submitted to the appropriate team (usually development/programming) for a feasibility review. Priority for the change was assigned based on the importance of the change to end users and the difficulty of implementation, and the accepted requirement was included in an addendum to the official specification. In rare cases, proposed changes or new requirements were determined to be out of scope, in which case they were listed in that section of the addendum so as not to lose the idea entirely.

Lessons Learned

Apart from adjustments to the schedule that occurred as the process unfolded, the project followed the plan for gathering requirements more or less as intended. Over the course of the requirements writing period, the team learned to work effectively at a distance. In addition,

several side benefits of the requirements specification became evident as the project progressed.

Working effectively at a distance

Since the project team was distributed over North America, certain challenges arose having to do with working remotely. Documents were exchanged in electronic form, and wherever possible, collaborative technologies were used to avoid duplication of effort. For example, the requirements were captured in a database to which everyone on the team had access so that the current listing of requirements was available at any time. Wikis were used by the development team to keep track of progress on requirements as well.

Regular conference calls were used to keep the participants in touch and aware of deadlines and progress. The team found that having a steady "pulse" for the project helped keep people on track even when other responsibilities intervened. The difficult work of prioritizing requirements, which involves balancing several factors for each requirement under consideration, was done in conference calls so that all opinions could be heard.

Side benefits of having user requirements

The team found that the process of collecting requirements was valuable in itself, because it provided a forum for everyone to think about and discuss what the end product would look like. In the beginning everyone had a different idea about what would be built; working through the process of capturing and refining requirements helped the team to come together with a shared vision.

Having the document to refer to during development helped keep the team focused. When the question arose of whether to adopt a new technology (APOLLO; please see "Architecting the Elephant: Software Architecture and User Interface Design for Pachyderm 2.0" for more details), the decision-making process was framed in terms of whether the functions added by APOLLO would conform to the requirements specification. A requirement-by-requirement review was conducted to assess whether adopting APOLLO would be an appropriate step.

The Take-Away

The process described here focuses on the user as the source for most of the requirements. It can be adapted to projects of various sizes, and because none of the methods involve great expense, it is appropriate for small teams as well as large ones. The key elements are a strong collection of source documents from which to pull requirements; a range of user-based studies (scenarios and observations in this case; focus groups, task analysis, or other methods may also be appropriate); a requirements collection process anchored with a simple format for recording requirements; and a thorough sorting, classifying, and prioritizing effort. The time involved is considerable, especially for large projects; but the end result is well worth the effort.