Pluto: Community-based News Server

by

Dennis A. Quan
S.B. Chemistry
Massachusetts Institute of Technology, 2000

Submitted to the Department of Electrical Engineering and Computer Science in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Computer Science

at the

Massachusetts Institute of Technology

January 2002

© 2001-2002 Dennis Quan. All Rights Reserved.

The author hereby grants to MIT permission to reproduce and to distribute publicly paper and electronic copies of this thesis document in whole or in part.

Signature of Author.......................................................................................................................................................... Department of Electrical Engineering and Computer Science

January 15, 2002

Certified by ........................................................................................................................................................................ Walter Bender
Senior Research Scientist
Thesis Supervisor

Accepted by ......................................................................................................................................................................... Arthur C. Smith
Chairman, Committee on Graduate Students
Department of Electrical Engineering and Computer Science
Pluto: Community-based News Server

by

Dennis A. Quan

Submitted to the Department of Electrical Engineering and Computer Science on January 15, 2002, in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Science

Abstract

Internet technology has enabled individuals to publish their opinions and create community dialog in an inexpensive fashion, but the barrier to entry is great for those who are not familiar with web servers and HTML. In addition, most web server environments do not facilitate traditional processes for peer revision. Pluto is a web server designed to address both of these problems, by providing users with an intuitive and easy-to-use environment that promotes editorial processes, automatically generates HTML, and manages the web site.

Thesis Supervisor: Walter Bender

Title: Senior Research Scientist
# Table of Contents

Acknowledgements.............................................................................................................................. 9

1 Introduction........................................................................................................................................ 11

1.1 Related work............................................................................................................................... 12

1.2 Document organization.............................................................................................................. 12

2 Pluto in use ....................................................................................................................................... 14

2.1 Usage scenario............................................................................................................................ 14

2.2 Melrose Mirror .......................................................................................................................... 16

2.3 Junior Journal ............................................................................................................................. 19

3 Theoretical framework...................................................................................................................... 22

3.1 Community-based news ............................................................................................................ 22

3.2 Editing process........................................................................................................................... 24

3.3 Usability....................................................................................................................................... 25

3.4 Extensibility.................................................................................................................................. 28

4 Design.............................................................................................................................................. 31

4.1 User management....................................................................................................................... 31

4.2 Articles....................................................................................................................................... 32

4.3 Baskets....................................................................................................................................... 34

4.4 Multimedia................................................................................................................................. 36

4.5 Sections....................................................................................................................................... 36

4.6 Version control............................................................................................................................ 37
# Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Melrose Mirror</td>
<td>17</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Junior Journal</td>
<td>20</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Home Base</td>
<td>27</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Pluto Tour</td>
<td>28</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Article Editor</td>
<td>33</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Group Basket</td>
<td>35</td>
</tr>
</tbody>
</table>
Acknowledgements

This thesis is dedicated to my mother, Nancy Quan, who gave me the inspiration to accomplish great things. Thanks for continuing to push me on.

I would like to thank Walter Bender for his support, mentorship, insight, and inspiration, without which Pluto could not have been realized.

Many thanks go to Ingeborg Endter for her patience in working with me to develop, document, and test the system. I would also like to thank Marko Turpeinen, Jack Driscoll, and Brian Smith for their input and their efforts in making Pluto a success.

I want to thank my friend Piotr Mitros for introducing me to this project and to Walter Bender and for his support during the development of Pluto.

Finally I am indebted to many students at the Media Laboratory, the Junior Journalists, and the Silver Stringers for their tolerance of my code and their feedback. In particular I would like to thank Spiros Tzelepis and Don Norris for giving me their feedback in the interviews I conducted with them.

This work was supported in part by IBM and the MIT Media Laboratory's News in the Future research consortium.
1 Introduction

In the past, news—condensed into newspapers, magazines, and broadcasts—was reported by large media companies for consumption by the masses. The means for giving feedback or engaging in dialog with these news producers were both simple and limited: avid readers could write letters or submit articles to their editors, but with no guarantee of their feedback being published or being read by the public. Individuals or small groups could create their own publications (this is after all permitted by the Constitution), but distribution to an audience larger than a local community was limited by the financial resources required to print and mail en masse. In short, the news media provided an inhospitable platform for community interaction and two-way dialog.

The past decade has seen the development of the Internet, lowering the barrier to entry for millions to have their voices heard. By putting up a website, individuals are now able to publish their opinions with very little expense. Beyond website creation and hosting costs, distribution is virtually free. There is one hidden cost however—the learning curve for this technology.

In a traditional newspaper system, journalists write or type up their articles and send them to editors, who then mark up the drafts with a pen and return them. Although the public basically already understands this process thoroughly—people experienced it when they had papers graded in grammar school—the equivalent electronic process is daunting to many. Even for those who can surmount this challenge, the final step of web publication, namely translation to HTML and posting to an HTTP server, serves to eliminate most of this group.

Pluto is an attempt to solve some of the problems mentioned above. Pluto is a web server designed specifically to allow those with little or no experience with a computer to publish articles electronically. In addition, Pluto facilitates the writing process, allowing authors and editors to revise articles prior to publication. Pluto also takes care of HTML generation and makes it easy for users to set up the organization of their web site. Finally, Pluto takes a step forward towards the idea of community-based news—a concept whereby small groups are able to put forth their opinions and truly achieve mass distribution.
1.1 Related work

Much work has already been done in the effort to help users express their thoughts online. Internet communities such as Geocities (Yahoo!) and theglobe.com provide online forums for user interaction. Visitors can post their thoughts on online discussion groups, and members (membership is usually free) create web content and/or manage a portion of a web “community.”

Commercially available software, such as IBM WebSphere Homepage Builder, Microsoft FrontPage, and Macromedia Dreamweaver, makes it easy to design web pages. These packages provide word processor-like functionality and design templates to assist users in developing HTML-based web content. They also provide tools to help users upload files to and manage files on web servers, giving users a single environment for producing, editing, and publishing various kinds of web sites.

Pluto, on the other hand, is designed specifically to facilitate the editorial process and to build community-based news sites. While Pluto is capable of hosting arbitrary web content, Pluto promotes human communication by enabling editors and authors to exchange articles and to engage in dialog before deciding to publish. Web page design tools can also be configured to work with an editorial cycle, but this often requires significant customization.

1.2 Document organization

Chapter 2 discusses real-life usage of Pluto. First I give a walkthrough of how authors and editors interact with Pluto to create, revise, and publish articles. Afterwards, two large user groups, the Silver Stringers, and the Junior Summit, are described, and an exploration of how these groups use Pluto to publish their online magazines is presented.

Chapter 3 discusses the concepts upon which Pluto is designed. Core to Pluto’s purpose is the idea of community-based news, as noted above. First I discuss the theoretical notion of community-based news. A rundown of traditional editing processes is presented and compared with the experience Pluto provides. I also elaborate on how Pluto addresses usability issues in light of the types of users Pluto is designed to support. Finally, I address the question of extensibility—how Pluto will grow to users’ needs as they make progress in
their savvy of the technology.

Chapter 4 describes the design of the Pluto server. I discuss the types of objects Pluto supports and how they are used to permit authors and editors to produce publications. In addition, I describe some of the infrastructure put in place to enforce security and provide extensibility.

Chapter 5 addresses the actual implementation of Pluto. I describe the benefits and downsides of using Java for a project such as this and discuss the construction of the key services of Pluto, such as the object store and the directory. Finally, I describe the process administrators use to install, configure, and maintain Pluto systems.

Chapter 6 gives an evaluation of Pluto based on feedback received from active users of Pluto. Finally, Chapter 7 gives some concluding remarks and presents some lessons learned during the course of Pluto’s development.
2 Pluto in use

2.1 Usage scenario

In order to motivate some of the goals I had in mind for building Pluto, I will introduce Pluto's usage paradigm with a simple fictitious example. The Plutonian Press is a small online community newspaper published by a group of 30 reporters and 7 editors. The newspaper is divided into four sections, News, Sports, Finance, and Leisure, and each section has an editor responsible for it. Two editors serve as copy editors, revising for content, grammar, and spelling. The final editor is the editor-in-chief, overseeing the entire publication. The reporters are loosely affiliated with one or more sections of the paper, and some are also photographers.

At the moment some work is being done on a set of articles that highlight Hawaii, one of the premiere tourist destinations on Earth. One reporter, Mary, is preparing to start work on one of the articles, entitled “Kona: more than just coffee”. She goes to the newspaper’s home page and clicks on Log In, a special link at the bottom of the page that allows her to enter the internal part of the publication. After typing in her user ID and password, she is presented with her Home Base—in essence, a homepage for each Pluto user, providing access to article creation, revision, and other features.

She clicks on My Basket and is taken to her personal basket. Baskets are used in Pluto to hold articles and other items, and each user is given a basket to hold their works in progress. Mary then clicks on Create new article and starts entering her text. After she writes as much as she can, she considers getting input from the Leisure editor. She types “What do you think?” in the Notes field and clicks Save; she then returns to her basket. She clicks the checkbox next to her new article that has now appeared in her basket, selects “Leisure” and clicks Send. This sends the article to the Leisure editorial basket, a special basket available to the editors of the Leisure section.

Later that day Mark, the Leisure editor, logs in and clicks on the Leisure basket option in his Home Base. He finds the new article and opens it. He thinks it needs a few pictures to spice it up. He appends a note back to Mary in the Notes field telling her to take a picture of a
coffee plant. Since the picture of a coffee plant may be useful to others, he tells her to put it into the Public Storage, a special basket that is available to everyone. After saving his changes and returning to the Leisure basket, he checks the box next to the article and clicks Send to Author.

The next day Mary finds that her article has been returned to her. She opens it up and examines the Notes field to see what she must do. She takes a picture of a coffee plant and scans it in to the computer. Now, from her basket, she clicks Upload image. After following the instructions on the page, she makes note of the fact that the “ID” assigned to the image is “1034”. Returning to her basket, she checks the newly uploaded image and clicks Send to public storage. Afterwards, she opens the article and appends her article body with the text “[image 1034]”. That instructs Pluto to embed the image at the bottom.

She makes a few more changes, and now she feels that the article is done. She saves it for the last time and sends it to the Copy Editors basket. One of the two copy editors reads the article from the Copy Editors basket, makes some minor changes, and then forwards the article on to the Leisure basket.

Mark now sees the Kona article in the Leisure basket along with four other articles on Hawaii. He decides that these articles deserve their own subsection for this issue. He clicks on Edit Sections in his Home Base, an option he has access to because he is an editor. This takes him to the Front Page, which lists the top level sections in the publication. He then clicks on Leisure to see the Leisure section. Pluto supports sections within sections with arbitrary depth. He clicks on Create new section to add a subsection and names it “Hawaii: Paradise on Earth”. Going to the Leisure basket, he opens the Kona article and assigns the article to the new Hawaii section. Additionally, he selects a Hawaiian-styled template for the article, complete with pictures of the beach and pineapples. Templates in Pluto are responsible for translating articles into HTML. Finally, after saving the article and returning to the Leisure basket, he clicks the checkbox next to the article and clicks Publish. This posts the article to the website.
2.2 Melrose Mirror

Although the scenario above is fictitious, the various steps used to move articles from conception to publication form the backbone for real-life usage of Pluto. In fact, the revision cycle for articles in Pluto was borne out of the processes already in place with the first group of users of Pluto, who call themselves the Silver Stringers.

Silver Stringers is a six-year-old research project being done at the MIT Media Laboratory. The basic idea is to enable seniors, people with both the interest to act as raconteurs and the stories to recount, to publish their thoughts on the Internet. (Silver refers to the hair color.) However, many seniors are not familiar with computer technology, and as such another goal of the project has been to make storytelling over the Internet accessible to these people.
Figure 1: Melrose Mirror

A group of seniors at the Milano Senior Center in Melrose, Massachusetts were enticed to become the first subjects for this ongoing experiment. They have been publishing their newspaper, the Melrose Mirror, since 1996.¹ At the beginning, they used custom software written at the Media Laboratory based on Jigsaw, a prototype web server written in Java by the World Wide Web Consortium, to help them publish. In the first two years of the project,

the Silver Stringers learned the basics of writing articles and sending them to editors electronically.

However, the system had many limitations. First, being built on Jigsaw, which at the time was very unstable, made system maintenance a chore. It was also difficult to transplant the software for use with other groups. Also, images were not supported directly; they had to be uploaded to the web server using traditional means (e.g., FTP).

Pluto was developed in 1997 to overcome these problems. In early 1998 the Silver Stringers were migrated to a Pluto server. Since then the publication has grown to well over a dozen sections with almost a thousand articles written by more than 30 users. The Stringers have also benefited from the support for multimedia in Pluto. A large fraction of their articles have pictures, and they have learned to use sound and video on their website.

Despite taking advantage of all of this technology they continue to meet every week in person. At these meetings they discuss ideas for writing new articles and evaluate the content and structure of the publication as a whole. Many of the Silver Stringers look forward to these meetings every week as they typify the sense of community these publications strive to build.

Not long after the Melrose group started publishing on the Internet, the idea of senior citizens producing an online community publication spread to the Jack Satter House, another senior complex in Revere, Massachusetts. They maintain their own site now called *Satter-lights.* 2 Similarly, a group of senior citizens in Danvers, Massachusetts have started their own site called *Danvers Oracle* at [http://danversoracle.media.mit.edu:4000/](http://danversoracle.media.mit.edu:4000/). These sites help to demonstrate the success of Pluto in facilitating its original target audience, senior citizens, in producing community-based online publications. However, Pluto has turned out to be a useful tool for other age groups as well.

---

2.3 *Junior Journal*

During the fall of 1998, the MIT Media Laboratory hosted the Junior Summit, a gathering of a hundred young delegates from around the world to discuss issues affecting children worldwide. The delegates ranged from elementary school students to high school students and had varying levels of exposure to computers. At the beginning of the summit the delegates divided into groups to focus on specific challenges. One of these groups set out to let the world know of their progress over the Internet. They established the *Junior Journal*, a monthly publication hosted at [http://journal.jrsummit.net/](http://journal.jrsummit.net/).  

---

Figure 2: Junior Journal

During the week of the summit, the founding members of the Junior Journal organized themselves and prepared to run the publication together even though after that week they would be in different countries, working in different time zones, and often writing in different languages. They decided to correspond completely using e-mail, designating a few of themselves as editors.

The Junior Journal is run using Pluto and has been published continuously since the summit. In this three year time period, their user base has grown to almost 150 users, and they have composed more than 1500 articles. Using the editorial basket system in Pluto has been essential to keeping the newspaper in sync because of the lack of face-to-face communication in this community.
The members of this community are in many ways different to those in the Melrose community. Besides the obvious age differences, the members in this group rarely meet face to face, unlike the Silver Stringers, who meet weekly. Articles in the Junior Journal are often written in one language and translated into English and other languages. Additionally, many of the Junior Journalists had extensive computer experience. Despite these differences, Pluto's framework allows both groups to effectively produce their publications and support their respective communities.
3 Theoretical framework

In this chapter I focus on the concepts that make Pluto successful as a tool for communities to publish their thoughts effectively and easily. I start by discussing how a publication can be used to encourage interaction within a community.

3.1 Community-based news

The idea of using a news publication as a medium for dialog is not new and has manifested itself in “Letters to the Editor” and Opinion columns in newspapers. This interaction between journalists and their readership base serves to help keep biases towards any one side of an issue in check. As the world begins to adopt the Internet as an efficient means of news distribution, news producers have in turn become more plentiful. At the same time, the need for interaction between the “writers” and the “readers” becomes even more important. With large, affluent newspapers, where reputations have been earned over decades of service, readers often feel a sense of trust in their editorial judgment. However, with websites literally springing up overnight, readers must learn to consume this outpour of information cum grano salis. Many sites host discussion groups to allow them to gauge the different perspectives in their user base as well as to give readers a better sense of the diversity of opinion. On many sites, within minutes of a new article, criticism and ovation alike are spooned out with great abandon. The important result here is that readers have become much more actively involved in the production and maintenance of information than ever before. Further, as readers become more involved in news production, the distinction between the information producer and consumer is blurred.4

Many sites refer to this phenomenon as “community”—a group of individuals who interact and exchange perspectives in some context. Traditionally, community can be defined in several ways. McMillan and Chavis report the four main characteristics of a community as

membership, influence, fulfillment, and emotional connection. Discussion groups hosted by web sites often encourage active participation by their prominent placement and quick response by the writers of the site. This helps to instill a feeling of influence to the user base. Membership and fulfillment are encouraged both by economic incentives (posting on a site is usually free and sometimes rewards are given for posting frequently) and by the facility with which users can post messages on the site. Emotional connection can be derived from the discussion group having a highly themed or political topic or just by the fact that the discussion group allows its users to socialize.

Pluto encourages a slightly different type of community. The community is not simply grown out of response to various articles on a site; instead, the focus is on building a community from the writers within the site itself. The writers are usually also readers of other sites. In some sense this represents a separation of the publications themselves and the discussion groups created to respond to those publications that were discussed above. However, in separating the two parts the opinions have been re-aggregated in a somewhat different fashion. Instead of viewpoints being centered on specific articles, they represent a community-centered response—a viewpoint that is in some ways shared by many members of a community. As members of a community discuss not only their feelings on a particular subject but also the commonalities between their various opinions, the aspect of emotional connection is strengthened.

In addition to actively responding to news on other sites, Pluto encourages other forms of dialog not often cultivated by the discussion groups on web sites. On a community news site, it is the community and its history that serve as the focal point. Community history has not often been well preserved for those communities that do not have newspapers to chronicle it. Further, the groups using Pluto often share another trait: an interest in learning about technology. Because the users' levels of experience with computers is so varied, the

---

process of becoming familiar with Pluto in itself leads to shared learning experiences, enhancing emotional connections and fulfillment.

3.2 Editing process

The article lifecycle in Pluto is patterned after well-developed editorial processes that have evolved at newspaper publishers over the past few centuries. The mechanics of these processes have changed markedly in the last hundred years as technology has progressed, but the basic notions of editing have remained the same.

Many newspaper publishers have adopted an editorial hierarchy consisting of various first-line editors that manage groups of reporters and successively higher level editors reporting ultimately to an editor in chief. The first-line assigning editors generally cover a specific geographic region or topic (called their “beat”) and decide what articles should be written. Reporters work on specific articles and revise them with their respective assigning editors. Meanwhile, those assigning editors act as advocates for the articles their reporters are working on and negotiate for space and prominence in the section or subsection of the paper managed by the second-line editors to which they report. These editors then take the articles their subordinates have been working on and negotiate for space in the sections controlled by their superior editors, and so on. Ultimately, the managing editor determines the organization of the entire news section, and the editor in chief orchestrates the organization of the entire paper.

After an article is completed by a reporter and his/her first-line editor, it is sent over to a copy editor. Copy editors edit the grammar and mechanics of articles in addition to managing page layout and writing headlines. Copy editors, like other editors, are also organized in a hierarchy. Typically there is a chief copy editor (sometimes called the “slot”) who distributes the workload to his/her regular copy editors. Copy editors usually follow some established communications protocol when there are questions concerning the content in an article. Often this involves copy editors conferring with the chief copy editor, the

---

assigning editor, and even the original reporter in order to resolve an issue. After the regular copy editor is satisfied with an article, it is sent back to the chief copy editor for final approval.

After the structure and content of a section have been determined, layout editors proceed to compose the actual pages by assembling the type and photographs. Often a group of copy editors, section editors, and other editors work together to realize the final production. In the past this work was done by hand on large sheets of paper. Today, most major newspapers are assembled with computer pagination systems, allowing editors to lay out pages electronically.

It is important to realize that no two newspapers have completely identical organization. Copy editors report to the managing editor in many newspapers, but in some companies copy editors are associated with different sections and report to section editors. Smaller newspapers typically have flatter hierarchies than larger ones. However, the basic elements of article lifecycle are the same in all publications.

Pluto supports these basic elements of article lifecycle: composition by a reporter, revision between the reporter and various editors, and ultimate selection for publication by an editor. However, Pluto does not prescribe any specific editorial organization. In this way Pluto can support many different editorial hierarchies customized to specific user bases.

In addition, Pluto abstracts the notion of page layout—typically a skill-intensive process on a computer—into pre-canned “templates” that lay out articles and sections in a standard fashion. Editors are still free to customize the templates to change page layouts, but for the majority of Pluto’s audience, Pluto takes care of layout issues and allows users to concentrate on managing article content. In fact, this compromise is typical of the many design choices made during Pluto’s development to give the user a productive experience.

### 3.3 Usability

Behind the design of any system is the careful balance between ease of use and power. Oversimplified interfaces often restrict the flexibility more advanced users want to “tweak” the system. On the opposite extreme, too many options and features can appear intimidating
to beginning users. Pluto is designed to assist people who may be new to computer technology in building online publications. This emphasizes Pluto’s tendency to err on the side of simplicity.

The first area in which Pluto tries to simplify the user experience is in the system paradigm. Using commonplace nomenclature and familiar processes for performing tasks in the system help users who are not familiar with computers get started quickly.

Pluto takes users to their Home Base when they first log in. Our experience with Pluto’s user base has shown that users feel comfortable having a place to which to go back when they want to start performing a task. This is essentially the same idea as the Start menu in Windows and the home page in the web browser.

To help users manipulate objects in the system, Pluto exposes objects including articles, baskets, and sections. The names were chosen to coincide with their physical equivalents; articles logically fall into sections, and baskets are commonly used for holding articles that are works-in-progress.

Throughout Pluto the user sees descriptions of the various options that are provided on each screen. For example, the Home Base screen gives users clear explanations of the areas accessible from the Home Base. The Help button on every page provides users with assistance from anywhere in the system. Tips are also sprinkled throughout the interface to help users discover non-obvious features and procedures in Pluto.
Additionally, to help new users become familiar with the system, a tour is provided that steps users through all of the important activities in Pluto, from article composition to uploading pictures.
3.4 Extensibility

In addition to making it easy for new users to work in Pluto, it is equally important for Pluto to sustain users' learning about computer technology. Many software tools directed at beginners are easy to use because they hide complex features and make assumptions for the user. Often these tools do not permit users to override these assumptions. Rather than being
a tool that users outgrow quickly, Pluto helps ambitious users discover advanced concepts, such as HTML formatting, page design, and web site organization. As the first computer application many of our initial users had experienced, Pluto is given the unique opportunity to introduce many aspects of Internet technology to these users. On the opposite end of the spectrum, many school environments have students who enjoy experimenting with systems, and Pluto gives these users the environment in which to do so. From personal experience I know that younger students like to explore systems, and having firm limitations in a system can be a hindrance to this exploratory tendency.

For many users, the first area of customization of interest is basic text formatting. Pluto supports standard HTML tags for specifying bold, italic, underline, as well as font and size. Early versions of Pluto exposed custom tags for formatting text, but this was retracted in favor of standard tags because the knowledge of standard tags would be useful and applicable to users in other environments. Similarly, users are free to embed images, multimedia, and even Java applets and Dynamic HTML (HTML that uses JavaScript to dynamically manipulate the document) using standard tags. (A helper tag is provided for images only because it is significantly simpler than the standard tag and takes care of generating the correct URL for the image.)

In addition to being able to format individual articles, users of Pluto have access to the templates that generate the HTML for all articles and sections. The use of templates in Pluto is discussed in more detail in Chapter 4. By embedding a powerful programming language such as Scheme, ambitious users are not limited to the choices made by the designers of the software.

Pluto also places no limits on the number or complexity of articles, images, sections, and discussion groups defined for a publication. While it was important to provide users with a usable “out-of-the-box” set of default settings, users can add, modify, organize, and delete sections to suit the organization of their publication. This is in contrast to some novice user software packages, which enforce a prescribed organization scheme in an effort to simplify the user experience. Because Pluto is as much about usability as it is about allowing users to learn about the system, placing inherent limitations on organization was not considered.
The final important area of customization is in controlling the editorial structure of the publication. Pluto provides a comprehensive yet easy-to-use user management facility from within Pluto itself. Users can be aggregated into one or more groups, and as mentioned earlier, each group is given an editorial basket. A system administrator is free to organize as fine-grained a hierarchy as desired to streamline the article lifecycle in his/her publication.
4 Design

Key to the design of Pluto are the four ideas discussed in the previous chapter. In this chapter I elaborate on the types of objects that are supported in Pluto and what roles they play in facilitating these ideas.

4.1 User management

In Pluto all users are assigned a user ID and a password in order to enforce basic security and maintain the identity of objects created by users. When a user attempts to access the Pluto system, he/she is first presented with a login screen. After entering his/her user ID and password the user is taken to his/her Home Base, a page that gives the user access to the system.

During initial testing with the Melrose community, some authors chose to compose their articles using word processing software and use one user ID to log in and enter those articles into the system. While Pluto does support this arrangement, especially for articles from guest authors from outside the community, I chose to keep different user accounts separate to allow users to maintain their own basket, which contains their own in-progress and to-be-revised articles, and to automate byline generation.

From the Home Base the user has access to all the functionality the user is privileged to access. (More will be said about this point below.) In particular, a user can access his/her profile to edit items such as his/her name, password, and email address, as well as provide a personal home page.

Just as in a newspaper, some users may be editors—people who advise, revise, and control content for the publication. In Pluto users may belong to one or more groups. Groups represent editorial bodies. For example, the set of groups in a Pluto system may include “Sports Editors”, “Feature Editors”, etc. By belonging to at least one group a user is considered to be an editor by the system. Conversely, not belonging to any group prevents a user from accessing the editor-only features of Pluto, such as configuring the sections in the publication or publishing articles to the site.
For the purpose of administering the system, Pluto maintains a special group called System Administrators. Although not technically editors, system administrators are granted the highest level of privilege, being able to add and remove users, edit content, and in general access all parts of the system.

From a traditional computer security perspective, this simplistic group paradigm tends to generalize the different “roles” users can play in the system. A policy-based security system could have been used in Pluto, allowing administrators to selectively allow certain users to publish without being able to edit the sections of the publication. However, the goal of Pluto is to foster community and the writing process for people who may not be highly experienced with computers. I believe the user management system in Pluto provides a good compromise between access control and simplicity of administration.

4.2 Articles

In their simplest forms newspapers and magazines are composed of collections of stories. Similarly, a Pluto publication is composed of articles. An article in Pluto contains the copy of a story as well as the photographs, videos and other multimedia items associated with it.

Articles are edited in Pluto from a standard web page form. Instead of requiring users to enter HTML, a skill that was not often found in the test user base described earlier, individual metadata fields are made editable, such as title, subtitle, body, etc. In order to translate an article’s plaintext into HTML, Pluto supports the concept of article templates. Article templates are scripts that translate articles into HTML and are responsible for the formatting and layout of a published article. (Templates are discussed in detail in section 4.11.) For advanced users, it is possible to create a template that simply passes the body of the article through to the published version, allowing users to write their articles in raw HTML.

The default template passes HTML tags through to the generated pages, allowing users who wish to learn HTML to do so slowly. One of the goals of Pluto is to help communities learn to use technology more effectively. By allowing users to practice with HTML tags Pluto helps users gain sophistication with web publishing at their own pace. The default template
also takes care of issues with HTML that users may not expect. For example, Pluto replaces
“<” and “>” with &lt; and &gt; , respectively, where it thinks these symbols are not
being used to represent HTML tags. Similarly, two carriage returns in sequence is translated
into a paragraph tag because beginning users are not likely to be familiar with HTML’s
whitespace handling rules.

Figure 5: Article Editor

Articles possess several basic fields that help the users of the system organize articles. The
title field allows the author to name the article. The author field provides a place for the
author to customize the byline of the article. This is useful for when articles are written by
more than one author. Pluto does not support creating articles from more than one user at a
time; the article must originate from a single user. The section field allows the author to
designate which section into which to place the article; it can be left unspecified up until
publication. The template ID field gives the user a choice over how he/she wants the article
to appear. Templates are configured by administrators and are discussed later in more detail.
The summary field is useful for providing a summary or subtitle for the article. The body
field is where the author enters the body of the article. Finally, the notes field is meant as a
scratchpad either for the individual user’s thoughts or as an area where editors can leave
comments for the author and vice versa.

4.3 Baskets

To facilitate the exchange of articles and other items between writers and editors, Pluto
provides containers called baskets. Baskets contain articles and items of various types
described below. Each user in Pluto is given a personal basket, and each group (except
System Administrators) is given a group basket.

Articles typically travel through many baskets before being published. An article originates in
the author’s personal basket. From there, the article can travel to a group basket, which
usually represents an editorial group. From a group basket, the article can be sent to other
group baskets, back to the author, or to publication.
Figure 6: Group Basket

Since each group (and hence each set of editors) has its own basket, a multistep editing process can also be instituted. For example, with groups named “Copy Editors”, “Photograph Editors”, and “Feature Editors”, an article can travel between any of these editors and the author before being published. However, Pluto does not support rigid workflow patterns; it is up to the users how they choose to route articles between editors before publication.
4.4 Multimedia

In addition to copy, articles found in modern publications often include photographs, pictures, diagrams, and with the introduction of the Web, video and sound clips. Pluto represents an individual picture or clip as a media item, which can be moved between baskets just as articles can.

Users import media items into Pluto from their baskets. (Pluto does not actually support the creation of such items; they must be produced by separate graphics, sound, or video editing software.) Once imported into the system media items are assigned an object identifier (OID). To place an image on a page a user need only type “[image oid]” into an article to have the image appear at that place in the text.

In fact, all objects on the system are assigned OIDs (this will be discussed in more detail later). OIDs make it easy for users to identify articles, images, and other files regardless of their location in the publication. When objects move from basket to basket or from section to section, their OIDs do not change.

A URL can be generated for an OID by using the notation “[link oid]” in an article. For example, if article 45 needs to have a hyperlink to a video with OID 65, article 45 can contain the following:

\[<a href=[link 65]>Click here to see the video</a>\]

This is another example of exposing real HTML to allow users access to more advanced web functionality.

4.5 Sections

Most newspapers are divided into sections, e.g., Front/News, Business, Sports, etc. These sections are often further divided into subsections, e.g., National, Weather, Politics, etc. In contrast, a magazine may or may not be split into sections. To permit this level of flexibility in organizing an online publication, Pluto supports sections within sections. At the root of the publication is one main section called the Front Page. All sections are contained within
this root section. In this way Pluto’s sections are akin to folders on a filesystem.

Like articles, sections have templates that produce the HTML viewable by readers. The default section template generates a page that displays a list of the subsections to the left and a list of the articles in the section on the right. The root section typically contains a special template that displays a list of selected articles from various sections in the publication that will appear on the front page of the publication. While this resembles the result discussed in Chapter 3 of the front page being a showplace of the various articles in a publication, these articles are not normally stored in the root section; instead, links to the articles are displayed, much like the cover of a magazine.

4.6 Version control

Many modern word processing applications, collaboration platforms, and software development tools provide a feature called version control. The basic idea is that authors often wish to revisit or return to old revisions of their work. Sometimes writers begin progressing on a document in a certain fashion only to discover that another approach is better. Many times older versions of the document will have used this alternate approach, but at some point the approach was abandoned; by revisiting an old draft writers can resume work along this track. Saving old versions is also useful for reflection on one’s progress in a document.

Pluto supports a version control system that allows authors to revisit previous versions of articles. Authors can also elect to “activate” some past draft, making future progress on the article based on this draft. However, during real life usage of Pluto, the version control system was not extensively explored and is a possible avenue for future research.

4.7 Public storage

Usually media items are assigned to the same section in which the article using the media item is. However, when an image is used by more than one article, perhaps in different sections, or if the image is used as a “logo” on every page, it becomes a problem deciding where to best store the image. During early user tests with Pluto there was confusion about
the proper place to keep items of this nature. To solve this Pluto provides a special basket called the Public Storage.

All users have access to the Public Storage from the Home Base. An item can be sent between the Public Storage and any basket on the system. Items in the Public Storage do not need to be published; references to images in the Public Storage from articles display correctly.

4.8 Discussions

Pluto includes support for discussions groups. These discussion groups can either be accessible by internal members only or by both members and the readership. Their purpose is to encourage feedback and to build an external community in the same way other web sites have as discussed in Chapter 3. Users can participate in discussion groups from their Home Bases.

Like articles, discussions are objects with an OID, live in sections, and have templates that control their presentation. Discussions also contain the set of messages that have been posted by users, and each message in turn has its own OID. A discussion object specifies the templates that are used to format the discussion, its messages, and a form for posting new messages as HTML to the user.

Messages in a discussion can also be mediated. A Boolean flag in the discussion group’s settings controls whether postings appear immediately in the discussion or require approval by the mediator first.

Unfortunately, this feature was not used extensively. I postulate that users of Pluto were more apt to use face-to-face communication to discuss topics and that feedback from external users usually came in the form of e-mail. A better means of incorporating discussion groups into this type of software is the subject of future research.

4.9 E-mail

Pluto has a built-in POP3 e-mail client accessible from users’ Home Bases. At the time when this feature was added e-mail was not widely available to people who did not have accounts
on major UNIX host systems. When Pluto was run in conjunction with an LDAP directory server and an LDAP-enabled mail server, e-mail accounts were easily created for all users on the system.

Towards the end of development however, the popularity of free web-based e-mail from sites including Hotmail (http://www.hotmail.com/) and Yahoo! (http://mail.yahoo.com/) made integrated e-mail less appealing. Additionally, purchasing and maintaining an LDAP server and mail server was prohibitive to many groups wishing to use Pluto to publish to the web economically. The current version has e-mail features hidden by default.

4.10 Archive

As a publication grows new content often replaces older content. The Archive is the centralized location in Pluto designed to hold content that is no longer being exposed to readers. An article becomes archived when an editor selects the article and clicks Send to Archive from the section where the article resides.

Because in Pluto, objects’ OIDs never change, any links in articles to archived articles continue to function. In this way Pluto is somewhat resistant to the broken link problem, a common ailment of web sites using traditional directory structures to store and refer to files.

The Archive can hold articles, HTML files, sections, and media items. Because the Archive is structured like a section, archived sections become subsections of the Archive.

4.11 Templates

Templates generate HTML representations for articles and sections in Pluto and allow users to concentrate on the content of their publication. By delegating the responsibility of creating HTML to templates, beginning users no longer need to know HTML in order to publish to the web.

The idea of separating design from content is not new. Cascading style sheets (CSS) is a standard way to separate the semantic markup in HTML (e.g. heading tags, definition tags, paragraph tags, etc.) from presentation markup (e.g. color, font face, font size, etc.).
Similarly, the Extensible Stylesheet Language (XSL) standard is a means for separating the presentation logic from XML-formatted content.

Templates are written in a combination of HTML and a subset of Scheme, a dialect of Lisp. The way in which HTML and Scheme are combined resembles the format used by Microsoft’s Active Server Pages (ASP) and Sun’s JavaServer Pages (JSP). In fact, keeping article content in a database and serving it to clients using ASPs or JSPs is a popular approach used by many news sites today. The body of the template is written in HTML with embedded tags that contain Scheme. The following special tags are used (these tags are very close, if not identical, to the corresponding tags used in ASP and JSP):

- `<% statement %>`: statement is executed at the point in the document where the tag is found.
- `<%= exp %>`: exp is evaluated and the value of the expression is inserted into the document at the point where the tag is found.

Details on the use of Scheme and the specific APIs employed to generate pages are presented in Chapter 5.
5 Implementation

In this chapter I discuss the Java-based infrastructure upon which Pluto is built. I also elaborate on how the various services in the system fit together to support the object hierarchy discussed in the previous chapter.

5.1 Java

Pluto was written completely in Java for the JDK 1.1 platform. Java was selected over other languages because of the following features:

1. Portability: Pluto can run on Windows 9x, Windows NT (and hence Windows 2000 and XP), MacOS, Linux, and other versions of UNIX. (In practice, Pluto is most often run on Windows NT.)

2. Garbage collection: because Scheme was used as Pluto’s scripting language, having garbage collection built into the platform made it convenient to support Scheme.

3. Object oriented features: writing Pluto in an object oriented language made it easier to maintain the modularity of Pluto’s design. In addition, the native support for object serialization made it easy to implement the database.

4. Network services: the Java platform provides support for many of the standards used by Pluto, including POP3, SMTP, LDAP, and HTTP.

5. Industry support: because of wide-reaching support for the language and the platform, I felt that there would be sufficient tools available for developing, testing, maintaining, and debugging Pluto.

Despite these advantages, the Java technology available at the time Pluto was written suffered from some major problems:

1. Instability: early Java 1.x virtual machines (JVMs) contained bugs that prevented Pluto from running continuously for long periods of time. Either the server would stop responding, accessing Pluto would become sluggish, or the virtual machine would completely consume system memory.
2. Slowness: early JVMs also lacked Just-In-Time (JIT) compilers that convert Java byte code into faster, more efficient native code.

3. Incompatibility: while the JVM found on Windows was found to be the most stable and fastest, stability and speed varied widely on other platforms.

At present most of these problems have been fixed in the Java 2 platform.

5.2 Dtypes

In designing a framework for data storage in Pluto, several requirements were considered. First, data objects needed to be easily stored and retrieved from disk. Second, data objects needed to be accessible from the script interpreter used for page templates. Third, the framework needed to be powerful enough to describe the structured data describing articles and other objects in the system, yet be extensible to permit end-user customization. Finally, to support inter-application data transfer, the format needed to be standardized.

A natural way to model data is to define classes in Java. Customized Java classes can be read and written from the filesystem via the Java Serialization API, and many Java-based scripting engines support the use of Java objects from script. However, data exchange is difficult because only other Java-based applications can easily read this format. Problems also arise when future versions introduce incompatible changes to the data model.

Another possibility was comma-separated files. While highly portable, they did not offer the rich typing support needed. Also, there are few standards on how comma-separated files should be formatted for a specific application. Currently, XML alleviates many of these difficulties, but this technology was not yet available at the time Pluto was developed.

Dtypes, invented at the MIT Media Laboratory, is essentially a binary encoding standard for standard Scheme data types.\(^7\) Scheme, being a dialect of Lisp, uses s-expression grammar to encode arbitrarily complex data structures and possesses a basic type system (such as vectors,

pairs, strings, Booleans, and integers). Pluto uses Dtypes to store article, section, template, and other data.

Specifically, the following is an example of an article described using Dtypes.

```plaintext
#((class . "Article")
  (title . "Pluto Features")
  ...
)
```

Objects in Pluto are represented as vectors of key-value pairs (by pair I am referring to a cons) called environments. Environments are keyed by arbitrary identifiers. Some identifiers have special meaning to Pluto's Object Store, such as class. All other identifiers are free for use by applications.

### 5.3 Directory service

Pluto relies on a directory service to authenticate users and to maintain session state. When a user logs into Pluto, the directory service checks the user ID and password and issues a ticket, a string that uniquely identifies that user's session. The ticket automatically expires after a fixed period of inactivity, helping to maintain the overall security of the system. This idea of a ticket or session cookie is commonly used by e-business web sites and security systems such as Kerberos.

Pluto comes with two directory service implementations. An LDAP-based directory service uses an LDAP server to store user information. LDAP, or Lightweight Directory Access Protocol, is a popular protocol supported by many popular directory servers, including Novell Directory Services, Microsoft Active Directory, Netscape Directory Server, and others. The idea of centralizing user information is becoming popular as users become inundated with multiple services (e.g., e-mail, calendaring, bank accounts, collaboration, etc.) all requiring separate logins. To the network administrator, keeping user information in sync between these multiple services is often a daunting task. When running Pluto in a large networked environment, it can be advantageous to have Pluto use an existing directory
server for authentication as well as profile information (including full name, e-mail address, etc.).

As an alternative, a leaner local directory service is also provided. The local service exposes the same interface as the LDAP-based directory service but uses a Hashtable to manage the user list and the Java Serialization API for persistence. Also, the local directory maintained by this service can only be used by one Pluto server at a time.

The following forms are exposed by the directory service:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(login username password)</td>
<td>Verifies the user’s credentials given by username and password and returns a ticket, representing the authenticated session.</td>
</tr>
<tr>
<td>(verify-ticket ticket)</td>
<td>Returns true if the given ticket is still valid; otherwise it returns false.</td>
</tr>
<tr>
<td>(getuserid ticket)</td>
<td>Returns the user ID associated with the ticket.</td>
</tr>
<tr>
<td>(logout ticket)</td>
<td>Discontinues the session represented by the given ticket and returns true.</td>
</tr>
<tr>
<td>(newuser ticket commonname username password firstname lastname)</td>
<td>Creates a new user with the given information (commonname refers to the user’s full name) and returns true.</td>
</tr>
<tr>
<td>(newgroup ticket groupname)</td>
<td>Creates a new group and returns true.</td>
</tr>
<tr>
<td>Function Call</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>(userinfo ticket)</code></td>
<td>Returns user information associated with a ticket, such as first name, last name, and the user ID, as an environment.</td>
</tr>
<tr>
<td><code>(querygroups ticket)</code></td>
<td>Returns a vector of all group names.</td>
</tr>
<tr>
<td><code>(querygroup ticket groupname)</code></td>
<td>Returns a vector of the user IDs present in groupname.</td>
</tr>
<tr>
<td><code>(is-user-in-group ticket username groupname)</code></td>
<td>Returns true if <code>username</code> is in <code>groupname</code>; otherwise, returns false.</td>
</tr>
<tr>
<td><code>(delete-user ticket username)</code></td>
<td>Deletes the user identified by <code>username</code> from the directory (has no effect on objects the user has created) and returns true.</td>
</tr>
<tr>
<td><code>(groupadduser ticket groupname username)</code></td>
<td>Adds <code>username</code> to the group identified by <code>groupname</code> and returns true.</td>
</tr>
<tr>
<td><code>(group-remove-user ticket groupname username)</code></td>
<td>Removes <code>username</code> from the group identified by <code>groupname</code> and returns true.</td>
</tr>
<tr>
<td><code>(deletegroup ticket groupname)</code></td>
<td>Deletes the group named <code>groupname</code> and returns true.</td>
</tr>
<tr>
<td><code>(queryusers ticket usernames fields)</code></td>
<td>Returns an environment associating the individual usernames in <code>usernames</code> with environments associating the fields in <code>fields</code> with the values corresponding to those fields for that user.</td>
</tr>
</tbody>
</table>
5.4 Scheme interpreter

The biggest motivation for using a variant of Scheme for scripting in Pluto is to provide easy access to Dtype-structured data. Because Dtype data types are the same as Scheme's data types, it is natural to use Scheme constructs to manipulate this data. Pluto implements a subset of the full Scheme language, and Pluto's implementation has some important differences:

- No support for floating point numbers. No floating point arithmetic was ever needed during development, so it was not implemented (neither was the division operator incidentally). However, this support would be very easy to add.
- No support for quasiquote.
- Dynamic scoping instead of static scoping. This is mainly due to the convenience of implementation. No scripting in Pluto was ever complicated enough to make this difference meaningful, although arguably a true Scheme implementation would have kept to static scoping.
- No continuations.
- No letrec.

---

• No mutation (e.g., set!).
• No tail recursive optimization.

The following Scheme forms are supported in Pluto:

• List functions: car, cdr, cons, length, list-tail, reverse, append, vector->list, list->vector, vector, null?

• String functions: string-append, number->string, string<?, string<=?, string>=?

• Date functions: date-date, date-month, date-year, date<?, date<=?, date>=?

• Arithmetic operators: +, *, -, =, <, >, <=, >=

• Environment functions: env-lookup, env-sort

• Control constructs: if, do, begin

• Boolean operators: boolean/or, boolean/and, not

• Definitions and functions: lambda, define, let

• Miscellaneous: quote

Adding forms to Pluto's Scheme interpreter is accomplished by registering objects that expose the DtForm interface.

**5.5 Object Store**

The Object Store provides a persistence mechanism for objects in Pluto. All objects are stored as Dtype environments and are identified by OID.

The Object Store also provides access control based on the access specification within objects. All interactions with the Object Store use a ticket provided by the directory service as the basis for checking access control.
The following forms are exposed by the Object Store:

<table>
<thead>
<tr>
<th>Form</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(get ticket id)</td>
<td>Returns the environment associated with id.</td>
</tr>
<tr>
<td>(update ticket id env)</td>
<td>Adds the name-value pairs in env to the existing object named by id and returns true.</td>
</tr>
<tr>
<td>(put ticket id env)</td>
<td>Replaces the contents of object id with env and returns true.</td>
</tr>
<tr>
<td>(delete ticket id)</td>
<td>Deletes the object named by id and returns true.</td>
</tr>
<tr>
<td>(save)</td>
<td>Saves the current state of the database to disk (mainly for administrative use).</td>
</tr>
<tr>
<td>(repair-database)</td>
<td>Performs an integrity check on the database (mainly for administrative use).</td>
</tr>
<tr>
<td>(create ticket classID preferredID)</td>
<td>Creates an object of class classID, assigns the object an ID of preferredID or a default one if preferredID is null, and returns the assigned ID.</td>
</tr>
<tr>
<td>(verify-permissions ticket id action)</td>
<td>Verifies that the user has permissions to perform action on the object named by id and throws an exception if there is insufficient permission.</td>
</tr>
<tr>
<td>(quickupdate ticket id field value)</td>
<td>Changes the value of field to value in the object named by id and returns true.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>(query ticket ids unused keys types)</td>
<td>Returns an environment associating the IDs in ids of one of the specified types to environments containing the requested keys.</td>
</tr>
<tr>
<td>(revert ticket id revision)</td>
<td>Replaces the contents of the object with the revision identified by revision and returns true.</td>
</tr>
<tr>
<td>(delete-revision ticket id revision)</td>
<td>Deletes the revision identified by revision and returns true.</td>
</tr>
<tr>
<td>(exists ticket id)</td>
<td>Returns true if id is in use; otherwise, returns false.</td>
</tr>
<tr>
<td>(all-IDs ticket)</td>
<td>Returns a vector containing all IDs in use.</td>
</tr>
<tr>
<td>(undelete ticket id)</td>
<td>Restores the specified object if it has not already been permanently deleted and returns true.</td>
</tr>
</tbody>
</table>

Pluto’s Object Store supports a weak form of inheritance. The class key can be used to associate an object with a class. When an object is created, pairs of the class object are copied into the initial environment, and the class key is associated with the original class object’s ID.

Mutations made against Pluto’s Object Store are performed serially. Because the web server is multithreaded, methods in the Object Store are made thread-safe using the Java synchronized keyword. However, the Object Store currently has no support for transactions.
If the Object Store were to be reimplemented using a relational database as a backend, transactional functionality would be easier to support.

The Dtype objects themselves are stored on the filesystem. When an object is retrieved or modified, it is loaded from disk and placed into the object cache. There is no write-back cache; all modifications are committed immediately. However, the object cache serves to reduce unnecessary disk activity involved with accessing frequently-used objects. The object cache has a finite size in order to allow Java’s garbage collector to release less frequently used objects.

Access control is maintained by a special key called security that can be present in every object in the system. The value of this key is an environment mapping action names (e.g., read and write for reading and writing the object to the database, respectively) to vectors of user IDs and group IDs specifying what users and groups are permitted to perform these actions. There is also a special entry permissible in this vector that specifies which users and groups are expressly forbidden from performing these actions. (Affirmative permission is checked after prohibitive permission.) This entry takes the form of a pair whose first element is the symbol not and whose second element is a vector of user IDs and group IDs to be forbidden. Further, two special user IDs have significance in this context. guest is the user ID assigned to users viewing the external publication. Also, everybody is used to denote the set of all users on the system, with the exception of guest. Finally, Pluto Administrators are always permitted to perform all actions on any object. For example, the following object permits everyone to read the object and all editors except bob and charlie to write to the object:

```
#((security . #((read . #("everyone"))

    (write . #("editors" (not . #("bob" "charlie")))))))
```

### 5.6 Object structure

Many environment keys have special meaning in Pluto. Some keys, such as class and security listed above, affect the way the object is stored and retrieved from the Object Store. Keys
that affect the structure of a publication in Pluto are described in this section.

The section tree in Pluto is maintained using a parent-child hierarchical structure. The children key is used to identify a vector of child object OIDs. The parent key is used to refer to the parent object OID. Articles and subsections are considered children of a parent section. This restricts articles and subsections to be contained within one unique section at a time, just as files can be in one folder at a time on a filesystem (excluding hard and soft links, which are not supported in Pluto’s notion of parent-child hierarchy).

The following forms are used to manipulate the parent-child structure:

<table>
<thead>
<tr>
<th>Form</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(add-child ticket parent child)</td>
<td>Adds child to the children list of parent and makes parent the parent of child; returns true.</td>
</tr>
<tr>
<td>(remove-child ticket parent child)</td>
<td>Removes child from the children list of parent and clears the parent key of child; returns true.</td>
</tr>
<tr>
<td>(delete-tree ticket id)</td>
<td>Deletes recursively the object represented by id as well as its children; returns true.</td>
</tr>
</tbody>
</table>

In contrast, membership in a basket is not about parent-child containment. Instead, an object can technically be in many baskets at once, although Pluto is written so that an article generally stays in at most one basket at a time. When articles are published, they are removed from the group basket from which they originated. However, as soon as a section is assigned for an article, that section stays the article’s parent, through the editing phase and into publication.

<table>
<thead>
<tr>
<th>Form</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(kill-basket-items ticket basketID id)</td>
<td>Removes the object identified by id from the given basket and returns true.</td>
</tr>
</tbody>
</table>
(get-basket ticket unused username)  Returns a vector of the items in the basket of the specified user.

(add-to-basket ticket basketID id)  Adds the object identified by id to the given basket and returns true.

Another important key is the visible key. The visible key is associated with a Boolean value that determines whether the object is visible to external viewers. Publication in Pluto is a simple operation of setting the visible key to true.

(make-visible ticket id)  Sets the visible key of the object to true and returns true.

(make-invisible ticket id)  Sets the visible key of the object to false and returns true.

### 5.7 HTML generation

The following forms are provided to aide HTML generation in templates:

(convert-to-html string)  Converts article body text into proper HTML by changing tabs into a series of nonbreaking spaces, double carriage returns into breaks, evaluates bracketed expressions (e.g. \[link oid\]), etc.; returns generated markup.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(compile-html ticket id)</td>
<td>Compiles the template with embedded <code>&lt;%</code> expressions identified by id into a Scheme program and stores it using the key compiled-code; returns true.</td>
</tr>
<tr>
<td>(generate-html ticket id parameters)</td>
<td>Returns an environment mapping content to a string containing the generated HTML for the specified object.</td>
</tr>
<tr>
<td>(generate-href id)</td>
<td>Returns a URL to the given object.</td>
</tr>
<tr>
<td>(generate-special-href place)</td>
<td>Returns a URL to a special location, such as the login page.</td>
</tr>
</tbody>
</table>

### 5.8 Other miscellaneous forms

In addition to the Object Store's API, Pluto provides the following helper functions:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(update-media-object-info ticket id)</td>
<td>Makes a record of the media items referred to by the specified object and returns true.</td>
</tr>
<tr>
<td>(get-personal-item ticket property username)</td>
<td>Returns the ID of the personal object specified for username, such as a basket or mailbox.</td>
</tr>
<tr>
<td>(send-message ticket id)</td>
<td>Sends the message identified by id and returns true.</td>
</tr>
<tr>
<td>(is-mail-installed)</td>
<td>Returns true when mail support is installed.</td>
</tr>
</tbody>
</table>
(retrieve-mail ticket) Retrieves mail from the mail server for the active user and returns true.

(get-mailbox ticket) Returns the ID of the mailbox of the active user.

(submit-note discussion note) Submits the specified note into the specified discussion and returns true.

5.9 Java Servlet API

The Java Servlet API provides a universal platform for building web server components. Java Servlets are like CGI scripts in that they can produce dynamic content in response to web server requests. The servlet API abstracts much of the details of HTTP away from the servlet while exposing a powerful HTTP object model for those applications that require it.

All of the web serving functionality in Pluto is built to the Java Servlet interface. This reduced the amount of basic web server functionality that had to be written and permitted the focus to be placed on the server model. The Java Servlet development kit comes with a test server for running servlets, and this server suits the purposes of Pluto since all content is produced by the Pluto servlet. Before settling on the test server, Jigsaw, the W3C’s Java-based experimental web server platform, was also tried but was found to be too unstable at the time.

5.10 Server configuration

The Pluto server consists of a service manager whose function is to manage independent components of the server called services. Services can be started and stopped independently and accept Scheme forms as commands. The Object Store, the directory service, the web server, and the command line are all services. The command line provides a Scheme interpreter prompt that allows convenient access to the other services.
As mentioned earlier, Pluto is most commonly hosted on a Windows-based server. Because beginning users are not often familiar with setting up sophisticated server software, a self-extracting installer was built for Pluto using InstallShield. By double-clicking on the installer, Pluto can be installed within seconds, and the software comes preconfigured with default templates and icons on the Start menu to make it easy for users to get started quickly.

5.11 Internationalization

The Internet is a global network, connecting people who often do not speak the same language, as was indicated in Chapter 2. As a result, most modern software projects are designed to not be heavily tied to any specific (human) language. Users feel most comfortable with software when they interact with it using the language of their choice. Java, like other platforms, supports the separation of string tables (files containing locale-specific user interface text) from code. This makes it easy for developers to concentrate on programming the “logic” of an application while leaving the job of localization to (human) language experts.

Pluto comes built-in with an English string table and, as a (comical) example, a Latin string table. To date Pluto has been translated into Thai, Spanish, Portuguese, and other languages. (Translating into Asian languages presented the greatest challenge in that the code had to be careful to preserve the multibyte encodings of the various character sets.)
6 Evaluation

The effectiveness of any system is best measured by its users. I conducted interviews with three people who were involved in Pluto-based publications and were in positions to observe how effective the system was for themselves and for others in their groups: Jack Driscoll, former editor-in-chief of the Boston Globe and visiting scholar at the Media Laboratory; Don Norris, the editor-in-chief of the Melrose Mirror, and Spiros Tzelepis, the editor-in-chief of the Junior Journal. Driscoll has been heavily involved from the beginning of the Silver Stringers project in organizing the hundreds of publications that are part of this project. Norris has been key in directing the Melrose group as well as in providing technical direction. Both Driscoll and Norris provided me with invaluable feedback during the development of Pluto. Tzelepis was a teenager when the Junior Journal was launched and continues to direct the publication over e-mail with the other Junior Journalists around the world. Their feedback agreed on several points, but the areas where their opinions differed reflects on the diversity of the Pluto user base. In this chapter I discuss the results of these interviews and draw some conclusions on the effectiveness of Pluto. (The original interview texts are given in the Appendix.)

6.1 Usability for beginning users

All of the people interviewed reported that beginning users had little trouble becoming familiar with Pluto quickly. The primary obstacle to getting started with Pluto appears to have been general familiarity with computers. Norris noted that “the children of the Junior Summit all were experienced computer users, and effectively learned to use the program in one day. … Once they (new Silver Stringers) become familiar with the computer, learning Pluto is relatively easy.”

The interviewees noted that when just starting out with Pluto, users need a certain amount of persistence and interest to learn the system. Norris commented on the Silver Stringers that “past 70 years, immediate memory fades, and therefore a second or third repetition is advisable. Herein lies the need for patience, skill and understanding by the teacher.”

Similarly, Tzelepis found with the Junior Journalists that “however stimulating and friendly a
system might be, nothing can be done without the personal interest of the user (this is a principle that applies to all things in my view).”

Helping users along this path appears to have been the ostensibly simple, easy to remember processes users go through to perform basic tasks in the system. Norris stated, “I found that a user with little computer experience gains confidence in the logical sequences of Pluto. That experience helps him/her to handle the harder experience of learning computer basics.” This feedback is encouraging in that the basket system used for managing articles was designed in response to the feedback from the Silver Stringers concerning the previous version of the software.

Additionally, the interviewees reported that Pluto’s abstraction of low-level HTML and details of web server functionality from users was effective for beginners. “The beginner can start without having to learn HTML or other programming stuff and he can do work without needing much computer knowledge,” said Tzelepis.

6.2 Pluto as a learning platform

As was stated in Chapter 3, making advanced features available in Pluto not only satisfies the needs of more sophisticated users but also provides learning opportunities for ambitious beginners. I am happy to report that the users of Pluto responded positively to having this added flexibility. Norris notes that “some (of the Silver Stringers) have been inspired to learn and use HTML code.” Tzelepis elaborates on his experience:

In 1998, I was a person who could use a computer quite well but my knowledge stopped there (Besides I didn’t have any motive to go further). Eventually and responding to the complexity of the system (that was naturally increasing as I went deeper into it) made me to start studying, motivated me to learn and finally reach a level not only to be able to use all the features and possibilities Pluto offers but also to design websites. I owe all this to my dealing with Pluto.
6.3 Online help

Interestingly, the online help in Pluto received mixed reviews. Tzelepis felt that for the Junior Journalists, “the online help system is indeed very helpful. Personally, I was helped very much by it in my first days with the software and that’s why I always strongly insist that new users read all the introduction tour before they do anything else.” Getting users started quickly is precisely what the online help and tour were designed for. In contrast, Norris reported that “there is difficulty with the Help section—but then most of us refuse to use Help in any program because it frequently makes matters more confusing—and small type is hard to read. We collectively feel that all Help writers are technicians, not teachers.” Driscoll, having been present when both groups learned to use the system, not surprisingly thought that the online help system was “somewhat” effective.

There are several possible explanations for this difference in opinion:

1. Being more familiar with the nature of computer software produced in the last decade, perhaps the Junior Journalists knew what to expect from the online help and therefore knew how to use it best.

2. It is possible that the illegibility of the small type to the Silver Stringers audience prevented any useful information from being conveyed.

6.4 Encouraging human interaction

One of the primary goals of the system was to encourage human interaction both on and off the computer. I discovered that one of the most powerful ways to foster communication is to provide areas where users can record their thoughts. One area where Pluto provides this is in the notes field in the article editor. Driscoll felt that Pluto is “especially effective in the editing phase, providing the notes mode that enables comments to go back and forth.”

Another of these areas is the Discussions page in Pluto. Unfortunately, while it was reported that the idea of discussion groups were useful, most used other software to host group discussions. “Perhaps the least used items are the discussion features because jrsummit mailing lists substituted them,” said Tzelepis.
Additionally, users have opportunities to interact with one another when learning about the system together. Tzelepis explained that for the Junior Journalists, “the whole process of asking and explaining that occurs when somebody is new to the software is an interaction that helps new users integrate in the online community.”

Complementing human interaction are the workflow processes articles travel through within the system. This is important because human interaction should not be created simply to overcome difficulties with the system. Norris states: “I am a retired journalist, having worked with both newspapers and magazines. Pluto does a good job in emulating (the publication production) process.” Driscoll notes: “The basket system was based on an office scenario that would provide baskets for in, out, mail, etc. It is a mechanism to easily facilitate the movement of material from one place to another without the need for oral or written communication.”

### 6.5 Areas for improvement

The one feature that users found most difficult to use was template editing. I attribute the difficulties to two specific areas: the lack of a visual template editor, and the choice of Scheme as the scripting language. The reason why a visual template editor was never provided was that at the time writing an HTML editor would have been about as great an undertaking as writing Pluto was! At present many componentized HTML editors exist, including Microsoft Internet Explorer and Mozilla, and these software packages should be evaluated when considering future development of Pluto. The choice of Scheme was connected to the choice of the underlying data model. I present more comments on this choice in Chapter 7.

Not surprisingly, users also reported that many features that have only recently become possible were missing in Pluto. “I think that the addition of the search feature and that of the streaming video would be enough to make the system more functional,” notes Tzelepis. Unfortunately streaming video was at its infancy when Pluto was being written, and features like these should be addressed in future work in this area.

The interviewees presented mixed views with respect to the security model in Pluto. Norris noted that “the editors enjoy the separation of access, into four levels.” The four levels he
refers to are reporters, editors, administrators, and people who write their articles external of Pluto. Tzelepis on the other hand feels “a renovation integrating more advanced features in administration (different profiles/rights for users of different levels of access)” would improve the system. Further, Driscoll notes that “low-level users don’t have read access to the material of others, nor do they know where their stories/images are once they have left their baskets.” In any future work on Pluto, it may be appropriate to maintain the current three classes of users (reporters, editors, and administrators) as defaults, but provide a higher level of flexibility in defining access control for those groups who deem it appropriate.


## 7 Future work

Throughout this paper I discussed different ways Pluto has been used to support community-based news, from modeling the workflow that has been successfully employed by newspapers for decades to presenting a clean and logical interface to accessing the features of Pluto. I also detailed the Scheme/Dtype-based framework upon which Pluto is built. In this chapter I discuss, in the context of the user feedback given in Chapter 6 as well as my own experience, possible areas for development for future work. I also examine specific design decisions that I would have made differently given the state of technology at the present.

### 7.1 Email/Discussions

Support for e-mail and discussions were included in Pluto because I had felt that integrating these features into Pluto would introduce new users to these powerful collaboration paradigms. From experience however, I have concluded that these features were not used significantly because of the prevalence of ISP-based and web-based e-mail (e.g., Hotmail) and mailing list software. I would advise in future work that these aspects of the system be forgone in favor of more powerful and inexpensive yet user friendly preexisting solutions.

Furthermore, during the development of Pluto I tried to ensure that Pluto would be capable of working with other systems, including LDAP directory servers and POP3 mail server. However, in the end I have learned that it is more important that software of this nature be self-contained, as I feel this is one element that contributed greatly to Pluto’s widespread distribution. Often directory, mail, database, and web servers either require expensive licenses or are relatively difficult to install and maintain. Including sufficient base functionality “in the box” was critical to Pluto’s success.

### 7.2 Scripting languages

Despite being the language used by MIT to teach freshmen elementary computer programming techniques, Scheme (in its rawest form) has experimentally been found to be
insufficient for providing extensibility in a tool such as Pluto. In my experience I would suggest two possibilities for future development.

The first possibility is to hide scripting altogether from users. With graphical template editing tools, it is possible to hide the underlying scripting language completely from users. However, more powerful features (such as Turing universality) may be difficult to provide from within graphical tools. This may also undermine one of my objectives of allowing the system to possess broad extensibility.

The other (not necessarily exclusive) possibility is to provide users with a choice of scripting languages. Some scripting languages are easier to start with than others. Also, ECMAScript (also referred to as JavaScript) is the standard means for scripting web pages, and by using ECMAScript as the scripting language users will be able to reuse this knowledge when dealing with other environments. Technologies such as IBM Research’s Bean Scripting Framework (BSF) allow various scripting interpreters to be used, including those not known to the author of the host environment at development time. At present, with a host of different freely available script interpreters available for the Java platform, from Python and ECMAScript to Scheme and various dialects of LISP, giving users flexibility and adapting to users’ individual experience levels can be done relatively easily.

### 7.3 XML

When Pluto was first designed the idea of a universal, extensible data format was just being established. The work on Dtypes at the Media Laboratory was relatively developed and therefore made a good choice for Pluto’s data model. Shortly after Pluto’s completion, XML (eXtensible Markup Language) emerged as a new contender in this space. A variant of SGML (Standard Generalized Markup Language), XML benefited from the tag structure found in HTML in addition to its freeform vocabulary and has become widely accepted. XML parsers exist for nearly every popular language and platform. Standards produced by the W3C and the industry based on XML include:

2. XML Schema Datatypes: standardized primitive datatypes for use within XML.

3. XML Namespaces: a means for distinguishing coincident tag names originating from different vocabularies.

4. SOAP (Simple Object Access Protocol): a protocol for performing remote procedure calls described in XML.

5. XSLT (eXtensible Stylesheet Language Transformation): a language for describing XML to XML transformations, useful for translating from one vocabulary to another.

These technologies have made XML the preferred way to exchange extensible data structures in a portable fashion.
References


Appendix: Interviews

Interview with Jack Driscoll

Usability

1. What level of computer user do you consider yourself to be?
   Average.

2. What parts of the system are easy to use?
   Writing, especially.

3. What parts of the system are difficult to use?
   Creating templates.

4. Is the system intuitive enough for a beginner to learn to use immediately?
   Yes, based on experience.

5. Is the online help in the system effective?
   Somewhat.

6. The system tries to model processes that go on during the production of a publication. Which of these processes were done by hand rather than with the tools in Pluto and why?

   The basket system was based on an office scenario that would provide baskets for in, out, mail, etc. It is a mechanism to easily facilitate the movement of material from one place to another without the need for oral or written communication.

Encouraging interaction

1. In what ways does the design of Pluto foster human interaction?
It's especially effective in the editing phase, providing the notes mode that enables comments to go back and forth.

2. In what ways does the design of Pluto hinder human interaction?

Low-level users don't have read access to the material of others, nor do they know where their stories/images are once they have left their baskets.

Extensibility

1. What features did you find to be most notably missing from Pluto?

It would have been nice to have a better tool imbedded for internal editing of text.

2. How easy is it do you feel to extend the basic functionality of the system?

Lots of opportunities, it would seem

3. What are the most important areas of extensibility in Pluto?

A system for interactivity among groups would be an important step. Easier ways to integrate sound and video also would be areas of extensibility. Templates for prioritizing stories in a section and a variety of layout templates also would enhance it.
Interview with Don Norris

Usability

1. What level of computer user do you consider yourself to be?

I've been using a computer since the Apple IIc in 1982; By using an online connection to Dow Jones' data base, I was able to do well enough in the stock market to retire in seven years. I am proficient with word processors, spreadsheets, data bases -- but I wouldn't dream of removing the cover of a computer. I have been the prime teacher of all the SilverStringer software programs.

2. What parts of the system are easy to use?

Pluto uses common sense moves. A leads to B, and B to C ... It is logical and orderly. If progress in learning the program is slow, the cause most likely is the fault of the teacher -- and we have had technicians who were not good teachers, in spite of their other skills. 

All six editors learned Pluto by using it, but on the other hand, there are a few Stringers who refuse to use Pluto -- or the computer, for that matter. They submit typed or hand-written copy.

Of those Stringers who learned the computer during the past five years of publication, 90 percent of them also use Pluto at least at membership (second) level. Some have been inspired to learn and use HTML code. Seventy-five percent of our membership now own a computer.

3. What parts of the system are difficult to use?

Understand that most of our members had to learn to operate a computer first, then tackle Pluto. The most difficult area is persuading a student that the machine is logical, that the student is the master, and the program will not embarrass him/her.

There is difficulty with the Help section -- but then most of us refuse to use Help in any program because it frequently makes matters more confusing -- and small type is hard to read. We collectively feel that all Help writers are technicians, not teachers.
Notice that The Mirror generally uses nothing smaller than a (relative) 14 point type; nevertheless, Pluto’s Help system is lacking.

There is nothing really difficult in Pluto. But I also find that, in using both the computer and Pluto, that the learning Stringer will often find a comfortable level and refuse to go farther. A few are ambitious and want to know what the editors do -- the third level. And all six editors are now administrators (fourth level)...

We also see new Stringers now using logic to solve a Pluto conundrum. Most problems seem to be after a user arrives at My Basket; there is some confusion in the difference between Creating An Article and Uploading a Picture -- but a few trials usually cements the ritual. Most of us learned the program by trial and error.

The editors had most difficulty dealing with the Front Page code. It was all strange to us, and required considerable study and some trial and error -- and a few crashes. Your predecessor, Marko Turpeinen, created information boxes that helped; and we, in turn, developed a 'publication schedule' that deals not only with Front Page code, but describes the publication process from beginning to end. We developed the program called "Publisher of the Month", in which each of the six editors takes a turn, twice a year. We also sought to involve the general membership in this process, with only very limited success. Most of the membership is satisfied to "let the editors do their thing".

The editors wanted to modify Front Page headlines with small illustrations -- and crashed the system, twice. We did figure out how to add and place photos (such as mug shots and corresponding captions) on the Front Page, but would like to add a common piece of artwork with each headline in a special section.

4. Is the system intuitive enough for a beginner to learn to use immediately?

Yes. But 'intuitive' depends on the skills and experience of the beginner. The children of the Junior Summit all were experienced computer users, and effectively learned to use the program in one day. The Seniors of the SilverStringers have to start back at kindergarten, so to speak. Once they become familiar with the computer, learning Pluto is relatively easy.

Also keep in mind that, past 70 years, immediate memory fades, and therefore a second or
third repetition is advisable. Herein lies the need for patience, skill and understanding by the teacher. Our average age is now 76 years.

5. Is the online help in the system effective?

No. Not for us. If there is a problem, it gets solved via email. No phones. All email. To me, Help is a fat manual, written by a professional writer. Unfortunately, in the commercial world, manuals are now online, and are written by technicians. As for Pluto Help, I read it years ago, commented on its brevity, and never referred to it again. My apologies if you have expanded it.

6. The system tries to model processes that go on during the production of a publication. Which of these processes were done by hand rather than with the tools in Pluto and why?

I am a retired journalist, having worked with both newspapers and magazines. Pluto does a good job in emulating that process.

There was some confusion between originating a story in My Basket -- that is, typing the story directly into the form -- or using any word processor and then copy-and-paste. It was a small matter of learning the computer and the program while being a productive writer. There are THREE challenges here: learn the computer, learn Pluto, and learn to write professionally. Eventually our members found what was comfortable; most went to Wordpad or Word, while a few continued to write directly in the program form.

The only recommendation I would have to this question is to develop a reliable voice-recognition program. We tried a couple, with disastrous results. It was a waste of money and time.

The editors enjoy the separation of access, into four levels. The system works very well, and the transition, or access between levels is only slightly inconvenient -- like having to return to My Basket in order to move to another mode. However, the Pluto system added considerable speed in most transactions; this had been a complaint under the old system since many of our computers were older and slower.

Encouraging interaction
1. In what ways does the design of Pluto foster human interaction?

It is logical. It is easy to learn. It is easy to memorize the sequences. Also its speed -- over and beyond the original SilverStringer versions -- was welcomed, and solved our early difficulties of waiting endlessly for basic systems to happen.

Ironically, I found that a user with little computer experience gains confidence in the logical sequences of Pluto. That experience helps him/her to handle the harder experience of learning computer basics. Our editor-teachers always ended each session on line with the Mirror, regardless of the individual student's level; it was therefore a show of positive productivity, of accomplishing something material, rather than memorizing mere sequences.

2. In what ways does the design of Pluto hinder human interaction?

In the transition from Goofy to Pluto, the Stringers provided a fairly long list of suggested modifications. Therefore, the implementation of Pluto was easy. Add to that cooperation between user and author, the completely new system you (Dennis) provided to speed operating time, and we were delighted. It was a major step forward.

Extensibility

1. What features did you find to be most notably missing from Pluto?

One: Somehow the archive disappeared during some crash -- and was never re-activated. Two years of work was lost. However, since the Stringers do not edit-out old stories, every article is available on-line -- subsequent to that '98 crash.

Two: Now, at the pending transition to the new edition called HDL, we see a potential problem in modifying five years of files to fit that new system.

Three: All Stringer versions failed to provide a Front Page online storage system. In early times, we saved it as hard copy, but found that inadequate. We then devised a system ourselves, which wasn't perfect -- and it was only recently implemented -- but we could now re-create online an old (recent) Front Page.

Four: Some writers complained about the lack of width of the writing box, which I
understand, is a function of the underlying Microsoft program. (We discovered a solution to that problem: In the editor's notes box below the editing box, we found if we did not insert a hard return, the main editing box expanded to fit that copy. This in itself created a scrolling problem, which we discovered, was solved with forced line breaks).

2. How easy is it do you feel to extend the basic functionality of the system?

We, the Stringers, have been modifying the layout almost continually since we started. Frequently we had to approach our student-advisor for instructions, or advisability, but generally what changes we wanted, we were able to institute. The Mirror is unique on the Web in its layout; You won't find many websites using such easy-to-see type, or using such logical, simple-but-very-effective, uncluttered layouts.

We point with pride to the growing list of new sister-sites around the world, all of which (I believe) started out with Pluto. This in itself speaks well of its functionality and its ease of production. At this point there are some 600 copies in use around the world -- some of which look like The Mirror, most of which have a totally new look; nevertheless, the functionality remains the same: Four levels of access.

3. What are the most important areas of extensibility in Pluto?

Tough question. But I don't see an end to the extensibility of Pluto. I see continuing versions being produced, but at present Pluto is being used in a wonder of applications: Corporate level, educational level, organizational level -- representing all ages from middle school children through the original SilverStringers -- that includes two or three members in their late eighties.
Interview with Spiros Tzelepis

Usability

1. What level of computer user do you consider yourself to be?

I consider myself a good computer user (though I am never satisfied with my knowledge and I always want something more). I have a good command of HTML 4.0 and have constructed several websites. I have also had some contact with Java in the past though I have never been patient enough to learn the commands seriously.

2. What parts of the system are easy to use?

In my opinion, Pluto is an amazing system in terms that it caters for all tastes, both for the advanced and for the beginner user. The beginner can start without having to learn HTML or other programming stuff and he can do work without needing much computer knowledge. Besides, the interface is quite friendly and easy-to-use for everyone. Also the experienced user will find out that he can apply his knowledge on the system to create amazing results without much effort.

3. What parts of the system are difficult to use?

As I mentioned above, the advanced user after one or two sessions will not be facing any problems at all. However, there are still some parts inaccessible to someone without computer knowledge such as template creation and management which is the most difficult part of the system. But I think that even the novice will start finding his way around with some help either from the software itself (the online help is very useful) or from someone else who is familiar with it.

4. Is the system intuitive enough for a beginner to learn to use immediately?

Basic things that immediately interest the beginner can be learned without much effort especially if the user himself tries to practise his skills on a frequent basis. Of course, the user must take the chances given to him so that he can use the system and grasp its philosophy. However, stimulating and friendly a system might be, nothing can be done without the
personal interest of the user (this is a principle that applies to all things in my view). At this point, I need to mention that if someone wants to progress, the system eventually will lead him to an advanced level of computer literacy if he responds to it and tries to expand his knowledge. I think that this is the most important thing I have gained out of my 3 years of experience with Pluto. In 1998, I was a person who could use a computer quite well but my knowledge stopped there (besides I didn’t have any motive to go further). Eventually and responding to the complexity of the system (that was naturally increasing as I went deeper into it) made me to start studying, motivated me to learn and finally reach a level not only to be able to use all the features and possibilities Pluto offers but also to design websites. I owe all this to my dealing with Pluto.

5. Is the online help in the system effective?

The online help system is indeed very helpful. Personally, I was helped very much by it in my first days with the software and that’s why I always strongly insist that new users read all the introduction tour before they do anything else. Together with their password and the welcome message, I am advising them to read the tour, apply what they learned and then come back to me with their queries. As far as I am concerned, I consider the online help a very important feature.

6. The system tries to model processes that go on during the production of a publication. Which of these processes were done by hand rather than with the tools in Pluto and why?

Few are the items we haven’t used so far. Perhaps the least used items are the discussion features because jrsummit mailing lists substituted them. Also, we didn’t use the personal webpage and email account features.

Encouraging interaction

1. In what ways does the design of Pluto foster human interaction?

I believe that the whole design of Pluto encourages interaction especially through the special discussion features which are useful. The Junior Journal had the jrsummit lists so I cannot give a first hand experience of how good they are in practice but theoretically if they are
used, they give chances for contact. Moreover, the whole process of asking and explaining that occurs when somebody is new to the software is an interaction that helps new users integrate in the online community.

2. **In what ways does the design of Pluto hinder human interaction?**

There is not much to say on this. I think that the software itself is good as far as interaction is concerned. The whole point is how the community will use it. If they exploit its capacities than there are not going to any problems.

**Extensibility**

1. **What features did you find to be most notably missing from Pluto?**

I have many times experienced the need to search for specific items with their ID number or with key words. There is no such feature and it is a great obstacle when you need to find missing or older pieces. Moreover, I think that the software should support a similar feature for the readers when they want to search something specific. This is not supported as far as I know. Also, I would like Pluto to give more chances to exploit video/audio clips in articles. It doesn’t support streaming video so we have to use other servers when there is such a need which can be great trouble.

2. **How easy is it do you feel to extend the basic functionality of the system?**

I don’t know if this is feasible on the technical side but I think that the addition of the search feature and that of the streaming video would be enough to make the system more functional.

3. **What are the most important areas of extensibility in Pluto?**

I think that the system is a good model of an online paper on this basis, a renovation integrating more advanced features in administration (different profiles/rights for users of different levels of access), in article handling such as search machines and better preview in HTML templates and a better storage and archive system (all this featuring the experienced gained with the use of the old ones) would make it a perfect software for a publication.