An Experiment in Collaboratively Developed Data Models
by
Jesse Koontz
Submitted to the Department of
Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Master of Engineering
In Electrical Engineering and Computer Science
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
August 2000
© Jesse Lenker Koontz 2000. All rights reserved.

The author hereby grants to M.I.T. permission to reproduce
and distribute publicly paper and electronic copies of this thesis
and to grant others the right to do so.
An Experiment in Collaboratively Developed Data Models
by
Jesse Koontz
Submitted to the Department of
Electrical Engineering and Computer Science
on August 2000, in partial fulfillment of the
requirements for the degree of
Master of Engineering
In Electrical Engineering and Computer Science
Abstract

This paper discusses an experiment in which an online community successfully collaborated on an extension to the data model of an online service. The Photo Database service provides an online medium for photographs and related photo information to photographers. Users customize the service by creating custom data fields that store additional information about their photographs. The service supports collaboration by allowing users to share customizations. By defining the method of collaboration to be the creation and use of custom data fields, contributions are limited to photographers who actively use their customizations. During the four months of the experiment, the community converged on a couple of common customizations of which one emerged as a candidate to be added to the data model. While a new data field emerged, it was not used by the more active members as they did not take part in the collaboration on custom data fields.

Thesis Supervisor. Harold Abelson
Title. Professor, Department of Electrical Engineering and Computer Science
Contents

Introduction ................................................................. 5
Metrics of Success .......................................................... 8
The Original Photo Database Service ........................... 10
System Architecture ..................................................... 12
Data Model Design ....................................................... 13
Site Design ................................................................. 15
Results ........................................................................ 25
Discussion .................................................................... 31
Future Work ................................................................ 35
Acknowledgements ...................................................... 38
Appendix .......................................................... 39
   A Photo Database Data Model ................................. 39
   B TCL and SQL Code for Results Data .................. 46
Bibliography ................................................................. 49
List of Figures

Figure 1: Screenshot of Database Browsing Page ........................................................ 16
Figure 2: Screenshot of User Workspace ................................................................. 17
Figure 3: Screenshot of Photo Upload Page ............................................................. 18
Figure 4: Screenshot of Equipment Add Page ................................................................. 19
Figure 5: Screenshot of Custom Field Page ................................................................. 20
Figure 6: Screenshot of Administrator Listing of Users ............................................... 22
Figure 7: Screenshot of Administration Summary Page ............................................. 23
Figure 8: Screenshot of Administration Camera Page ................................................ 24
Figure 9: Screenshot of Administration User Summary Page .................................. 25
Figure 10: Number of Users of the Photo Database During the Experiment .............. 26
Figure 11: Average Number of New Users Per Day ................................................... 27
Figure 12: Number of Photographs in the Photo Database ....................................... 28
Figure 13: Average Number of Photograph Uploads Per Day .................................. 29
Figure 14: Custom Data Fields Added to the Photo Database .................................... 30
Figure 15: Number of Cameras in the Photo Database .............................................. 31
Introduction

Online communities provide a unique opportunity for large numbers of individuals separated in space and time to interact, share knowledge, and collaborate [Gre99a]. This paper discusses an experiment in which an online community collaboratively developed an extension to the data model of an online service. The service, Photo Database [Pho00a], was extended to support per-user custom data fields and sharing of these customizations. The new service was then installed on the web site of a large online community of photographers. The goal was to support collaboration on new data fields for the service. During the four months of the experiment, the community converged on a couple of common customizations. One data field emerged as a candidate to be added to the Photo Database data model.

The Photo Database service was originally a term project for the Massachusetts Institute of Technology Class 6.916. The service allowed users to store photographs with detailed information such as a caption, exposure date, camera, and film used. It contained a simple custom data field feature that allowed users to store additional information with each photograph.

While a goal of this experiment is to record changes in the service made by the community, it will also inevitably include changes in the community as a result of new features in the service. In *Users and Customizable Software: A Co-Adaptive Phenomenon*, Mackay suggests that customizable software makes for a good study of this
co-adaptive process because the customizations serve as artifacts that can be traced through the history of the project [Mac90].

In Mackay's study of adoption and customization of the X-windows system in project Athena [Mac90], the success of the collaboration between users was attributed to the fact that the community studied shared a common purpose: customize the UI of the X-windows environment. Similarly, the users of the Photo Database share the common purpose of presenting data with their photos.

As Mackay noted [Mac90], services in which users share customizations reduce learning time and maintain consistency. Mackay concluded that several key features are helpful in supporting sharing of customizations. First, it was suggested that services should provide a way for feedback to get from end users to software designers. As a centralized service, the Photo Database provides statistics about its long-term use. Second, services should allow users to capture the context in which their customizations were useful. The Photo Database was extended to keep a user's description of their custom data field's purpose. Third, a service should allow users to share customizations across the service. The Photo Database was extended to provide a summary of other users' active custom data fields with the user provided description. Finally, a service should provide dimensions along which to customize the service that match the users' tasks. The Photo Database is focused on providing data with photographs. Allowing users to collaborate on what data is stored with their photographs allows them to control the main part of the service.
By defining the method of collaboration to be the creation and use of custom data fields, contributions are limited to photographers who actively use their customizations. This method of collaboration does not support active discussion about how the service should be extended or what data fields should be added or removed. However, the goal was build a service that was responsive to users who actually use the service. The collaboration on what data fields to add to the service occurs over a longer time period, but results in useful data fields. Similarly, it may become clear over time that the community no longer uses some of the default data fields. By not using a data field, a user also contributes to the collaboration on what default data fields to remove from the service.

After identifying which custom fields had been recognized as useful by virtue of being adopted by the community, the service’s main data model could be extended to include the new data fields as part of the default information stored with each photo. By becoming a core data field, users will be able to use that information in the core features. Core features include searching the database and sorting a folder of photos.

To provide interaction, knowledge, and collaboration, an online community needs contributing users. The service was introduced to the existing community of photographers at http://photo.net [Phn00] on October 1, 1999, and monitored periodically until February 1, 2000. This is a fairly large community with over a 1000 daily returning users. They are also a community with a large number of photographs, interest in online community-based services, and a well-developed sense of how to organize their photographic work. They will select more precise standards for representing information
(camera models, film types, camera lenses, etc.). Further, the community exists to share knowledge about photography and would be likely to accept other users' ideas for custom data fields. It is expected that the community will adopt the service since there is a dearth of online tools for saving photos and information online [Gre99a].

**Metrics of Success**

The goal of this experiment was to encourage users to collaborate on custom data fields. As Mackay found that the more advanced members of the group were more likely to produce original customizations, [Mac90], it is expected that active users will suggest the useful custom data fields. Active users are more likely to spend the extra time populating an additional data field with each photograph upload. Further, the service maintains a significant amount of data about the users' camera and film equipment. It was expected that this equipment data would only be added by users who have adopted the service for long-term use.

To answer the question about how successful the experiment was, information was first collected on the number of users of the service, the number of photos uploaded, the number and type of custom fields added, and the number of cameras registered with the service. Then a qualitative measure of the number of active users and the level of adoption of the service by the community could be made.

The first metric of successful adoption of the service is the number of users. Before the service shows collaboration on new data fields, there must be enough users to collaborate
on extending the data model. Active users are defined as those who have uploaded photographs in two or more sessions.

The second metric of successful collaboration is the number of photographs uploaded. Users must have uploaded several photographs with custom data before a custom data field could be considered “adopted” by a user. The first and second metrics of a successful experiment indicate a level to which users have adopted the service. This would appear in the data as an increase in the ratio of photograph uploads to new users per day.

The main measure of success for this project is whether the community collaborates on and adopts a new data type. Adoption of a new data type means that several active users have created similar custom data fields and populated them with data. The data type must be useful enough to be made a default data field for each photo.

It was expected that users would add “Lens Type” as the next default data field. Most advanced photographers who keep track of film and camera information would likely keep track of the type of camera lens used to take the photograph. It is expected that active users of http://photo.net would also be advanced photographers who would have use for a lens type data field.
The fourth metric of success in the experiment is the adoption of the equipment data collection. Users who have adopted the service are more likely to take the time to enter additional data on the equipment they use.

The http://photo.net site can serve as a guide for estimating the number of users, photographs, and custom data fields required to reach the goal of collaboration on custom data fields. http://photo.net has tens of thousands of registered and thousands of returning users. Of those thousands of users returning, hundreds post new content [Gre99a]. Because several users will need to create similar custom data fields before one can be selected and some users will create fields that are not useful to other users, it is hypothesized that one in 10 custom data fields suggested will be useful candidates for adding to the core data model of the service. To reach a goal of a couple of new core data fields, it is estimated that there would need to be at least ten custom data fields, one hundred active users, 1,000 registered users, and thousands of photographs.

The Original Photo Database Service

The primary goal of the Photo Database is to provide an appropriate medium for online photographs. In Philip and Alex’s Guide to Web Publishing, Philip Greenspun [Gre99a] outlines the problem created by a digital photograph being separated from a physical artifact. The digital photograph lacks a connection to the photographer, the context from which it was created, and other information not captured in the pixels themselves. A digital photograph file copied from location to location becomes separated from that information. To solve that problem, he proposes a service available to the Web that
provides a Uniform Resource Locator (URL) to link a graphic to the larger body of information not available in the graphic format. The service provides three sizes of the photo, including the full size, a medium size thumbnail, and a thumbnail no larger than 200x200 pixels. The information URL provides a listing of the photograph’s information: the photographer, the camera, the film, a caption, and copyright permissions.

This centralized location of information has a maintenance advantage over embedding the information into the graphics file. Copies of a graphics file with embedded information would not be kept up to date with the changes made to the original. The centralized service also has an advantage over a client-side application. A centralized service allows users to collaborate. Users can learn from other users’ contributions (e.g. comments on photographs and presentations). This collaboration allows the photographs to grow in content beyond that of a client-side application or a graphics file with embedded information. Finally, the maintainer of a centralized service can monitor, support, and extend the features based on how the service is used. A large number of similar user customizations may indicate that the software should be extended.

The secondary goal of the service is to support a community of photographers. The service supports sharing photos through presentations, searching, custom data fields, and eventually data field value maintenance. Presentations are collections of a photographer’s photos with annotations directed to a specified audience of other community members. Searching is a browsing tool for finding information across text fields and other main data fields of public photos. Custom data fields allow users to save extra information with
their photographs. As users share ideas on custom data, they collaborate on what to adopt as new standard data fields. Eventually, some users can be selected to maintain the list of possible values for the standard data fields (e.g. the list of camera models and film types).

**System Architecture**

The Photo Database was constructed as a module for the ArsDigita Community System (ACS) [Ars00]. The ACS provides a community-oriented model for online services. Information for each of the photos and users is being stored in a Relational Database Management System (RDBMS) that is being dynamically queried to build Hyper Text Markup Language (HTML) pages [Hyp00]. This provides an encapsulation of the service at the HTML level. The RDBMS used is Oracle running on HPUX [Gre99b]. The web server is AOLServer configured to connect to Oracle and process the scripting language TCL [Ame95].

Many of the software systems used for this project were chosen because the ACS server architecture already uses them. However, each has an advantage for this project. It made installing the service on http://photo.net simple as that web site uses the ACS. The ACS is both a set of web development tools and also a set of conventions and examples. The Photo Database uses the user contributions, user registration, and general comments modules. It also uses the HTML formatting and TCL scripting conventions of the ACS. Oracle provides the reliability and scalability required to build a large online database backed dynamic web service [Gre99a]. Although it has some differences from the ANSI standard SQL [Ora00], porting the Photo Database to another RDBMS would not be
difficult if it supported transactions with atomicity, concurrency, isolation, and durability. TCL provides a simple, interpreted language with which to build service functionality. The simplicity of TCL code is important to future work on this module as it allows others to easily read and understand the Photo Database code.

Data Model Design

As a module of the ACS, the Photo Database has a set of SQL DDL statements that create the necessary tables, sequences, and indexes in the Oracle database (see appendix A). The data model follows the ACS convention for a module’s SQL file. The data model has two main parts. First, there are tables that store the main photograph information and the cameras and films registered with the site and used with the photographs. Second, there are tables that store the custom data fields for each user and the custom data values for each photograph. After a new data field has been “adopted” by the community, it would be manually added to the ph_photos table and would follow the data structure of the camera and film tables.

The main table of the Photo Database, ph_photos, stores information about each photograph. This includes information that users cannot control: user_id, the foreign key reference to the row in the ACS users table which describes the owner of the photograph; creation_date, which contains the date and time of creation of the photograph; and a few columns for storing the number and size of the thumbnails. The ph_photos table also has columns that users can control including the copyright information, caption, date of exposure, a flag to signify the photograph (and data) is viewable by the public, and the
foreign key references to the film and the camera the photograph used. Foreign key references to the film and camera tables (ph_cameras and ph_films respectively) are used in the ph_photos table instead of a plain text field to allow users to maintain a separate list of their equipment. This “normalized” table structure also supports searching for one particular camera or type of camera without concern about an anomalous spelling of the name of the camera for one photograph. Users create entries in the camera and film tables while the service administrator maintains the normalized tables that describe the camera models and film types (ph_camera_models and ph_film_types). The camera and film tables also include a creation_date column to allow the use of the service to be recorded over time. Finally, the graphic itself is optional. Users can create a photograph without uploading a graphic if they just want to store the photograph’s information.

The custom data fields are stored in a small meta-data system. The table ph_custom_data_fields stores the id of the user who created the field, the date it was created, and a description of what the user intended to save with the field. For each user who creates a custom data field, a meta-data table is created to store the values of his custom data field for each of his photographs. As the user creates additional custom data fields, additional columns are added to his meta-data table. No delete operation was implemented to prevent losing data about the use of the custom data fields that are no longer required. Instead an active/inactive flag was added to the ph_custom_data_fields table. Custom data fields are allowed to be dates, text (up to 4000 characters), numbers, and integers. The data field description and support for the dates data type was added for the experiment.
Site Design

The service was originally developed for adopted users. It was assumed that the interface should be useful to long-time users. For this project, the site was extended to support collaboration and monitoring of the experiment. In general, the pages were modified to emphasize the presence of custom data fields. An administration interface was added to display statistics on the metrics of success as a collaborative service.

The Photo Database module has three main user scenarios. Everyone has access to the public photographs and presentations through the search and browse interfaces (see figure 1). Registered users can visit their portfolio of photographs, folders, presentations, equipment, and custom fields. Module administrators can view recent photo uploads and summaries of system use at the administration interface. The TCL code for the Photo Database is published at the project information page [Pho00b].
The main page of the service is the user's workspace or "portfolio," (see figure 2). The portfolio was chosen as the most important display, and therefore as the first page of the web site, because it provides the most useful links for a photographer. The portfolio is a collection of a user's folders (Rolls, PhotoCD's) of photos, equipment, presentations, and custom data fields. It also provides links to create new folders, presentations, and custom data fields.
Before a photograph can be uploaded (see figure 3), the user must register a camera and perhaps a film with the site. Registering a camera involves selecting a camera model and providing any other available information about serial number or camera model variations for that camera. Registering a film involves selecting a film type and entering the
manufacturer and description. Although this makes uploading a photograph much harder, it encourages users to fill in camera and film data. This is not expected to reduce the acceptance of the site as it is anticipated that only active users would take part creating custom data fields. And active users would have enough interest in the service to spend time entering data on their camera and film equipment.

**Add Photo**

![Add Photo User Interface](image)

If this form doesn't serve your needs, try customizing this system.

- **File to upload**: Add this photo
- **Description**
- **User specified Photo key**
- **Exposure date**: December 23 1999
- **Lens**
- **Camera**: EOS-5
- **Film**: 1600, 400 Superia, 64T 1, (RTPI), 800 (for press professionals)

**Figure 3: Screenshot of Photo Upload Page**

jkootz@mit.edu
Camera Models and Films

Select the Camera Model of your camera to add it to the list of Your Equipment. Don't see your model... add a new camera model or add a new film type.

Camera Models

- Calumet
  - C-1, 8x10 View Camera
  - Cadet 4x5

- Canon
  - 10S (10)
  - A2
  - A50 Zoom (Digital)
  - AE-1

Films

- (non-film)
  - Compact Flash Card
  - Digital (non-film)
  - Digital Camera
  - digital film

- Agfa
  - Agfachrome
  - Agfachrome CT-18 (non-E6 actually)
  - Agfachrome CT-21 (non-E6 actually)

Figure 4: Screenshot of Equipment Add Page

The user interface of a recently added data field would likely mirror the camera and film interface. The data values for the “Len Type” field would be entered through the equipment section (see figure 4). The photo upload page would gain an additional pull down select box. The search page would have an additional select list to control complex searches.

The Photo Database was modified to place links to custom data field functionality as prominently as possible to draw the attention of the users. First, the summary of a user’s custom data fields was placed at the top of a user’s contributions to the Photo Database. Second, the photograph upload page includes a comment and link suggesting that
additional data could be saved with the creation of a new custom data field. Finally, the
link to the page for managing custom data fields was placed toward the top of the portfolio.

Custom Fields

Your Workspace : Photo Database → Custom Fields

Your Custom Fields

Add a Custom Field

* Lens (text, activated)
  The type (and manufacturer) of lens used for the photo.

Other Custom Fields

Here are examples of what other people are doing with custom fields

* Stereo Format: L, R, Parallel, Cross, Anaglyph (text) Identifies type of stereo image (Left image of stereo pair, Right image of stereo pair, Parallel-view monolithc image, Cross-eye view, red/green anaglyph image, or non-stereo)
* Lens (text)
* Lens (text) lens used
* Lens (text) Lens used
* Lens (text) Lens Information
* Lens Used (text) Storing information as to which lens I used while taking this photograph
* Comments (text) Public comments on photo’s
* Lens (text) Lens used for shot
* Comments (text)

Figure 5: Screenshot of Custom Field Page

The custom data field management page, shown in figure 5, has options for creating,
editing, and inactivating/activating fields. It also includes a list of all active fields created
by other users with a description of why they created the field. This was intended to allow
users to learn about useful data fields from other users, see which data fields were most popular and well defined, and create similar fields for themselves.

For example, a user creates a custom data field to store the type of lens used to take a photograph. The description “Saving the lens used to take photo” is entered when creating the field. Other users who view the custom field page of their portfolio see this and other custom data fields created by other users. After a large number of “lens type” fields are created, new users learn that “lens type” is a useful field especially after they see photographs with lens type data.

An alternative approach to collaboratively developing the data model would have been to actively request suggestions for new data fields from the community. This was not chosen because the goal was to find the data fields that were useful enough to have data values added for each photograph and selected by a majority of the active community. Another approach to solicit data field suggestions would have been to create a discussion forum. Although the photo.net community actively uses the discussion forum to discuss photography-related questions, this method was not chosen because the goal was to build a service not a discussion about a service. With the collaboration method used in this experiment, voting for a new data field consists of creating a custom data field and populating it with data. This limits the collaboration to users who have online photographs and data. These are users who are more likely to be knowledgeable about what data fields are useful.
The Photo Database was also extended with administration pages to provide reports on the use of the service. The main administration page, shown in figure 7, displays statistics on the most recent activity, total numbers of rows in the main tables, and links to reports of each section of the service. Sections include folders, cameras, films, presentations, users, and custom data fields. The administration interface includes pages for maintaining camera, camera model, film, and film type data. It also includes links to remove photographs, folders, and presentations. Photo Database users are photo.net users who have uploaded at least one photograph.

**Photo Database Users**

<table>
<thead>
<tr>
<th>Name</th>
<th>Photos, last day</th>
<th>Photos, last week</th>
<th>Average Upload bytes</th>
<th>Total Upload bytes</th>
<th>Disk Space Used (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roger Cortesi</td>
<td>164</td>
<td>13</td>
<td>2</td>
<td>295366</td>
<td>48440084</td>
</tr>
<tr>
<td>Marka Buchberger</td>
<td>105</td>
<td>8</td>
<td>8</td>
<td>386954</td>
<td>40630222</td>
</tr>
<tr>
<td>Jeff Wilson</td>
<td>6</td>
<td>77</td>
<td>8</td>
<td>87853</td>
<td>6764685</td>
</tr>
<tr>
<td>Stephen Sherrod</td>
<td>70</td>
<td>3</td>
<td>3</td>
<td>86342</td>
<td>6043950</td>
</tr>
<tr>
<td>Tower Leung</td>
<td>67</td>
<td>4</td>
<td>3</td>
<td>183490</td>
<td>12293846</td>
</tr>
<tr>
<td>Randy Beggs</td>
<td>65</td>
<td>1</td>
<td>1</td>
<td>249274</td>
<td>16202833</td>
</tr>
<tr>
<td>Steve c</td>
<td>61</td>
<td>15</td>
<td>15</td>
<td>253667</td>
<td>15473696</td>
</tr>
</tbody>
</table>

Figure 6: Screenshot of Administrator Listing of Users

jkootz@mit.edu
Figure 7: Screenshot of Administration Summary Page
Figure 8: Screenshot of Administration Camera Page

Philip Greenspun [Gre99a] found that user contribution summaries allow the system administrator to quickly assess who is active and the relative quality of their contributions. The administration pages include a user contribution summary designed to indicate how active the user is and how they are using their custom data fields, (see figure 9).
Photo Database Summary: Stephen Sherrod

Your Workspace  Photo Database Administration  One User

- Email: sfs@ipa.net
- Average File Upload size: 86342 (bytes)
- Total File Upload size: 6043950 (bytes)
- Number of Photos: 70

Custom Fields

The user has not created any custom data fields for their photographs.

Folders

- Kids, Cuter Than Yours (Folder) - 10 photos
- Single Photos (Folder) - 44 photos
- Solar Eclipse And Misc. Astrophotos (Folder) - 16 photos

Presentations

- Astrophotos by Stephen Sherrod
- Western Scenes Here and There
- Kids, Mine are Cute, yours aren't

Cameras

- Camera III -- Canon EOS Elan II (EOS-50)
- FTb -- Canon FTb
- T-90 -- Canon T-90 (800mm and 1200mm fixed aperture telephotos)
- Camera II -- Canon T-90 (800mm and 1200mm fixed aperture telephotos)
- Camera I -- Canon T-90 (800mm and 1200mm fixed aperture telephotos)
- Canon T-90, Various FD Lenses and Telescopes, -- Canon T90

jkoonst@mit.edu

Figure 9: Screenshot of Administration User Summary Page

Results

Before conclusions can be drawn about the success of the collaboration during the experiment, data must be collected for the four metrics of success. The Oracle database was queried for data on the use of the Photo Database (see appendix B). The data was
intended to answer questions about the rate of adoption of the service by the users and the number and creation date of the custom data fields. The totals of users, photographs, cameras, and custom fields were queried four times a month for each of the four months in the experiment. Comparing totals after the four months to intended totals from the metrics of success section shows roughly how well the system was adopted. At the end of the experiment, those totals were 170 users, 20 active users, 1982 photos, 16 custom fields, 220 camera models, and 490 cameras.

The first metric of success is the number of users of the Photo Database. Users are defined as registered members of the http://photo.net site who have uploaded at least one photo. Figure 10 shows the number of users over the course of the experiment. The sample frequency is four times a month.

![Users of the Photo Database](image)

**Figure 10: Number of Users of the Photo Database During the Experiment**
A more sophisticated measure of adoption is shown by comparing the rate of increase of new users to the rate of increase of new photographs. Figure 11 shows a four-day average of the number of new users at each day during the experiment. The four-day average was chosen to show the day-to-day volatility without cluttering the graph. There is a linear regression line plotted with the data to show a linear model of new users per day. Over the entire experiment, the site starts with 1.1 new users per day and ends with 1.6 new users per day.

![Average Number of New Users](image)

**Figure 11: Average Number of New Users Per Day**

The second metric of success is the number of photographs in the Photo Database. Photographs counted are defined as non-deleted rows in the photograph table (ph_photos). No effort was made to remove incomplete or duplicate uploads from the count as they represented a small percentage (<1%) of the total. Figure 12 shows the
number of photographs in the photo database. As with the first graph, the sample frequency is four times a month.

![Photographs in the Photo Database](image)

**Figure 12: Number of Photographs in the Photo Database**

To compare against the rate of increase of new users, the rate of increase of new photographs is depicted in figure 13. It shows the four-day average of the number of new photos at each day during the experiment. As with figure 11, there is a linear regression line plotted with the data to show a linear model of the new photographs per day. At the beginning of the experiment, around 10 photographs were uploaded each day. By the end of the experiment, the average number of daily uploads increased to 21 photographs.
Figure 13: Average Number of Photograph Uploads Per Day

The third metric of success is how well the custom data fields were adopted by the community. Parts of that metric are the number and type of custom data fields. Before a custom data field could be added as a default data field to the service, the number of common custom data types used by the community needs to be measured. Figure 14 shows how many custom fields were created, when they were created, and what type of data the field was storing. The number of custom fields was the sum of both active and inactive data fields. Two common types of custom fields emerged. The “Feedback” fields intended to save feedback from other users. The “Lens Type” fields stored information about the lens used to take the photograph. There were a few other custom data fields created that did not fit into either the “Feedback” or “Lens Type” category. The “Season”, “URL”, and “Stereo Image” custom fields were not similar to any of the other custom data fields.
To complete the picture of how well the custom data fields were adopted by the community, the user contribution summaries of the users who created custom fields were examined. Measuring the level of adoption by the number of photographs uploaded, the users who created custom fields ranked near the bottom of photograph uploads among the users of the service. There were even fewer users who consistently filled data into their custom fields.

The final metric of adoption of the service is the use of the preexisting data fields. Figure 15 shows the number of cameras registered in the service over the course of the experiment. The sample frequency is four times per month.
Discussion

The ultimate goal of the project was to see a new custom data field emerge and be used by a majority of the active users. Of the users who did create custom fields, few populated the fields with data. While a new field emerged, it was not used by the active members as they did not take part in the collaboration on custom data fields.

The final number of users, photos, and custom data fields fell short of the anticipated totals. Instead of 1000 users with 100 active users, the service had 170 users with 20 active users. However, the percentage of active users to total users did match the expected value of one in 10. Even with fewer users, the service doubled the expected number of photographs per user. With a total of nearly 2000 photographs, there were roughly 10 times as many photographs as users. While the total number of 16 custom data fields...
exceeded the anticipated two custom data fields expected for only 20 active users, only one custom field was identified as a potential new default data field. The “lens type” field was the largest group of custom data fields and their descriptions match almost exactly. It is likely that there were not enough adopting users to produce the two new custom data fields expected for the experiment.

From the perspective of these totals, the Photo Database was a successful experiment in a collaboratively developed data model. The overriding assumption of this experiment is that custom data field values would only be entered by active users who have adopted the service and are willing to invest time in adding data to their photographs. The number of users, photographs, custom fields, and cameras indicates that users are adopting or starting to adopt the service. The average number of new photographs each day increased faster than the number of new users. Figure 11 indicates that the average number of new users per day increased only slightly over the course of the experiment. Figure 13 indicates that the average number of new photographs per day doubled over the course of the experiment. Either some of the old users were continuing to upload photographs or new users were uploading a greater number than before.

The experiment was successful in helping users collaborate on custom data fields. Figure 14 shows that a short period of time after the first “Lens Type” and “Feedback” data fields were created, other users started creating “Lens Type” and “Feedback” data fields. Their self-provided descriptions for the expected use of their custom fields closely matched each other. It is very likely that the placement of custom fields in a user’s public
profile and in the user’s workspace helps to bring attention to the service-wide listing of custom data fields. This allowed users to learn about the other users’ data fields and create similar fields for themselves.

However, from the perspective of the number of photos that actually have associated custom field data, the experiment up to this point has not been successful. The low number of photographs with "Lens Type" data may indicate that the users have not found the field useful. It is possible that the reason better results did not emerge was that the interface for entering data and adding custom data fields was too cumbersome. It is also possible that data was not entered because these users had not "adopted" the service. It was expected that only active users would have adopted the service and spent the extra time required to populate the data fields.

A surprise result was that there are no active users who have created custom data fields. "Lens Type" was expected to be created and used by the active users because they are more likely to be advanced photographers with various camera lenses for which the service did not provide a default data field. One explanation of the low number of active users creating "Lens Type" data fields is that a large number of the photographs are from digital cameras. Digital cameras produce photograph files that are easier to upload and digital cameras are less likely to have interchangeable lenses. Users with digital cameras are likely to upload more photographs and have less of a need for a "Lens Type" field. Another explanation would be that insufficient time has passed for users to adopt the service.
The “Feedback” custom data fields are another example of users learning from each other. While these data fields were common and the descriptions very similar, the fields were not used by any of the users who created them. The explanation is simple. Custom data fields store information that the owner of the photograph uploads. Other users who view those photographs cannot add data to that field. Most of the “Feedback” data fields were deactivated when users realized that no other user would be able to provide “Feedback” in that manner. This serves as further evidence that users who created custom data fields were not active users. The photograph display page uses the ACS “General Comments” module to collect feedback for each photograph. A user who was not active on the http://photo.net site would not have been familiar with this existing feedback mechanism and therefore would have been more likely to create a “Feedback” data field.

There were several other custom data fields that were not copied by any other user. Separate users created the “Season”, “URL”, and “Stereo Image” fields. These fields had the highest percentage of populated data. However, these users had relatively few photos and are therefore not considered active users. This reduces the significance of their more thorough use of their custom data fields.

While the “Lens Type” data field was not adopted by the active users, it appears it would be used if it had been created as a standard data field. The standard data field for cameras was well populated by the users of the service. As mentioned in the site design, registering a camera in the service requires extra effort, as it is a separate process from
uploading a photograph. And before users can upload photographs, at least one camera must be registered. Figure 15 shows that the number of cameras remained almost three times the number of users during the experiment. This indicates that users entered more data than was required and took advantage of the standard data fields.

**Future Work**

This experiment serves as an example of how to support a collaboratively developed data model. The changes made to the Photo Database service demonstrate a user interface for soliciting user contributions, supporting collaboration, and summarizing user activity. Without the emphasis on custom data fields and the summary of user-provided explanations, the community would not have shared ideas as quickly. The experiment also serves as an example of how to establish a set of criteria for evaluating what is an “active” user and when an “adopted” custom data field should become a standard field.

The next step with this experiment is to see if the “Lens Type” custom data field is adopted in the same way camera type was after it is converted to a standard data field. The field would be added to the data model and user interface in the same manner as camera. A metric for successful adoption of the new field would be that both new photographs and existing photographs have values entered for “Lens Type”. Its adoption would indicate that the data field was useful and that the community successfully extended the service to better suit its needs. Since the number of photographs with values for a “Lens Type” custom data field remains small, it is not yet clear if the community would adopt a “Lens Type” standard data field. To better test if active users would adopt
the new field, it should not initially be a required data field, unlike the camera and film fields.

The full measure of success for an online collaborative experiment would be several years of use and development as a photograph service. The Photo Database service should continue to be monitored over time to watch the trends of use. Another new metric of successful collaboration on changes to the data model would be the continued use of the custom data fields to identify another standard data field candidate. Some of the methods of analyzing the results proved useful and should be added to the administration reports: the number of active users; the custom fields created by active users; and the number of photographs with custom data field values. The service should also be monitored for default data fields that are no longer used. A new metric would have to be explored for evaluating when a default data field was no longer used by the community.

The Photo Database could be rebuilt using a complete meta-data system. A meta-data system would be able to support more data types than the four hard coded in the TCL code used to implement the custom data fields. A meta-data system would be easier to extend with new data types. Additionally, it could be used to display the standard data fields, which would make adding (and removing) standard data fields easier.

A feature that would increase the quality of knowledge in the system is community moderated data fields. As the users are the experts in the system, their knowledge of the possible camera and film data values is more accurate than those of the programmer.
tasked with maintaining the software. The service would need different levels of user control over the standard data value tables. For example, anyone would be able to add a new camera, suggest a new camera model, and create a film. "Moderators" would be able to approve a new camera model. "Service-maintainers" would be allowed to select who could be a "moderator."

With a quantitative measure of an "active" user and an "adopted" custom data field, a meta-data based service, and community-moderated data fields, the service could become a self-correcting software system. It would be able to identify when the active users have converged on a new data field, add it to the standard data fields, and allow the community to maintain the data set. In addition, the service could identify data fields that were not being used by the community and remove them from default set of data fields. The more difficult challenge is in defining the quantitative measure of an "active" user, a new data field "candidate", and a field that was no longer used. This paper only in part answered those questions of defining active users and data field candidates.
Acknowledgements

First, I would like to thank Doctor Philip Greenspun for suggesting and supervising this project. In addition, I would like to thank Hal Abelson for his strong feedback and for making this project possible.

I also owe thanks to the members of the 6.916 (spring 1999) Photo DB Project team for their efforts in developing the Photo DB service. Student contributors to the code were Jesse Koontz, Josh Weaver, Richard Su, Jie Zhang. The project advisor was Jin Choi.
Appendix

A Photo Database Data Model

--
-- photodb.sql
--
-- jkoontz@mit.edu
--

-- data model for photo management service
-- written by Group 4, 6.916, 3/4/99
-- edited jkoontz@mit.edu Oct, 1999

-- create an administration group for photo database administration

begin
  administration_group_add ('Photo Database Staff', 'photodb', NULL, 'f',
  '/photodb/admin/');
end;
/

create sequence ph_folder_type_id_sequence start with 1 increment by 1;

create table ph_folder_types (folder_type_id integer not null primary key,
  folder_type varchar(50)
);

insert into ph_folder_types (folder_type_id, folder_type) values (ph_folder_type_id_sequence.nextval, 'Roll');
insert into ph_folder_types (folder_type_id, folder_type) values (ph_folder_type_id_sequence.nextval, 'PhotoCD');
insert into ph_folder_types (folder_type_id, folder_type) values (ph_folder_type_id_sequence.nextval, 'Folder');

create sequence ph_folder_id_sequence start with 1 increment by 1;

create table ph_folders (folder_id integer not null primary key,
  user_id integer not null references users,
  title varchar(50),
  photo_cd_id varchar(50),
  folder_type_id integer not null references ph_folder_types)
create index ph_folders_by_user_id_idx on ph_folders(user_id);

-- The following table is for user preferences

create table ph_user_preferences (  
  user_id integer not null references users(user_id),  
  images_public_p char(1) check (images_public_p in ('t','f')),  
  photos_sort_by varchar(100),  
  default_image_size char(1) check (default_image_size in ('s','m','l')),  
  prefer_text_p char(1) check (prefer_text_p in ('t','f'))
);

create index ph_userprefs_by_user_idx on ph_user_preferences(user_id);

create sequence ph_user_prefs_byuser_idx on ph_user_preferences(user_id);

create table ph_cameramodels (  
  cameramodelid integer not null primary key,  
  manufacturer varchar(50), -- e.g., 'Nikon'  
  model varchar(50), -- e.g., '8008/F801'  
  variation varchar(50), -- e.g., 'titanium'  
  last_modified_date date,  
  last_modifying_user references users,  
  modified_ip_address varchar(20)
);

-- to facilitate captioning photos with tech info, we keep track of which  
-- cameras each photographer owns (if they want to give us this info)

create sequence ph_cameraid_sequence start with 1 increment by 1;

create table ph_cameras (  
  cameraid integer not null primary key,  
  userid integer not null references users,  
  cameramodelid integer references phcameramodels,  
  prettyname varchar(50), -- e.g., "EOS-5 with date back"  
  serialnumber varchar(50),  
  date_purchased date,  
  creation_date date,  
  n_failures integer,
  date_surveyed date
);

-- the numbers below are just up until  
-- date surveyed; they are not kept up to date automatically  
-- as users enter rolls  
-- numbers below are just up until
n_rolls_exposed integer, -- "n sheets" for a view camera
purchased_new_p char(1) check (purchased_new_p in ('t', 'f'))
);

create index ph_cameras_by_user_idx on ph_cameras(user_id);
create index ph_cameras_by_model_idx on ph_cameras(camera_model_id);

-- we go to all this trouble because we want to be able to ask "Show
-- me all the Tri-X photos"

create sequence ph_film_type_id_sequence start with 1 increment by 1;

create table ph_film_types (    
    film_type_id integer not null primary key,
    film_type varchar(20) -- e.g., c6, k14, c41-bw, bw
);

insert into ph_film_types (film_type_id, film_type) values
(ph_film_type_id_sequence.nextval, 'Digital');

insert into ph_film_types (film_type_id, film_type) values
(ph_film_type_id_sequence.nextval, 'Black/White');

insert into ph_film_types (film_type_id, film_type) values
(ph_film_type_id_sequence.nextval, 'C41 (Color Negative)');

insert into ph_film_types (film_type_id, film_type) values
(ph_film_type_id_sequence.nextval, 'E6 (Color Slide)');

insert into ph_film_types (film_type_id, film_type) values
(ph_film_type_id_sequence.nextval, 'K14 (Kodachrome)');

insert into ph_film_types (film_type_id, film_type) values
(ph_film_type_id_sequence.nextval, 'Infrared');

create sequence ph_film_id_sequence start with 1 increment by 1;

create table ph_films (    
    film_id integer not null primary key,
    film_type_id integer not null references ph_film_types,
    manufacturer varchar(50), -- e.g., Kodak, Fuji, Ilford
    full_name varchar(50), -- e.g., Ektachrome Professional Plus
    abbrev varchar(10), -- e.g., EPP, RDP, VPS
    last_modified_date date,
    last_modifying_user references users,
modified_ip_address varchar2(20)
);

insert into ph_films (film_id, film_type_id, manufacturer, full_name, abbrev)
values
(ph_film_id_sequence.nextval, 1, '(none)', 'Digital', 'Digital');

create index ph_films_by_type_idx on ph_films(film_type_id);

-- The following table is for custom fields tracking
-- It allows us to do smart searching on fields, figure out
-- which fields are active, and which are "deleted" (since
-- we can't REALLY delete fields from Oracle).
-- (Now that we are use 8i we can. 8/11/1999)

create sequence ph_custom_field_id_sequence start with 1 increment by 1;

create table ph_custom_photo_fields (custom_field_id integer not null primary key,
user_id integer not null references users(user_id),
field_name varchar(200),
field_pretty_name varchar(200),
field_type varchar(200),
date_added date,
field_active_p char(1) check (field_active_p in ('t','f')),
field_comment varchar2(4000))
);

create index ph_custom_fields_by_user_idx on ph_custom_photo_fields(user_id);
create index ph_custom_fields_by_active_idx on ph_custom_photo_fields(field_active_p);

-- A table ph_user_(user_id)_custom_info will be created to store custom photo
-- info.
-- It's columns include photo_id, data field's (being add on)

create sequence ph_photo_id_sequence start with 1 increment by 1;

create table ph_photos (photo_id integer not null primary key,
user_id integer not null references users,
folder_id integer not null references ph_folders,
-- Can this photo be seen in the community
photo_public_p char(1) check (photo_public_p in ('t','f')),
camera_id integer not null references ph_cameras,
film_id integer references ph_films,
file_extension varchar(10), -- e.g. .jpg .gif
size_available_sm char(1) check (size_available_sm in ('t','f')),
size_available_md char(1) check (size_available_md in ('t','f')),
size_available_lg char(1) check (size_available_lg in ('t','f')),
-- These are the sizes of the thumbnails. It allows the client
-- to display the whole page even if the thumbnails are not yet loaded
sm_width integer,
sm_height integer,
md_width integer,
md_height integer,
lg_width integer,
lg_height integer,
photo_cd_id integer,
orphan_key varchar(50),
exposure_date date,
caption varchar(4000),
techn_details varchar(4000), -- f-stop, shutter speed, film used
-- If a recognizable person is in the photo, is the
-- model_release info available
model_release_p char(1) check (model_release_p in ('t','f')),
-- rights grants -- we do this in six separate columns so that we can
-- use an Oracle bitmap index to make queries faster
rights_personal_web_p char(1) check (rights_personal_web_p in ('t','f')),
rights_personal_print_p char(1) check (rights_personal_print_p in ('t','f')),
rights_nonprofit_web_p char(1) check (rights_nonprofit_web_p in ('t','f')),
rights_nonprofit_print_p char(1) check (rights_nonprofit_print_p in ('t','f')),
rights_comm_web_p char(1) check (rights_comm_web_p in ('t','f')),
rights_comm_print_p char(1) check (rights_comm_print_p in ('t','f')),
-- copyright statement is an HTML fragment, if they want to
-- fundamentally refer people to their Web server, they can have
-- a simple sentence with a hyperlink
copyright_statement varchar(4000),
file_size number,
creation_date date,
publisher_favorite_p char(1) default 'f' check (publisher_favorite_p in ('t','f'))
);

create index ph_photos_by_user_idx on ph_photos(user_id);
create index ph_photos_by_folder_idx on ph_photos(folder_id);
create index ph_photos_by_public_p_idx on ph_photos(photo_public_p);
create index ph_photos_by_m_release_idx on ph_photos(model_release_p);
create index ph_photos_by_r_pers_web_idx on ph_photos(rights_personal_web_p);
create index ph_photos_by_r_pers_print_idx on ph_photos(rights_personal_print_p);
create index ph_photos_by_r_nonp_web_idx on ph_photos(rights_nonprofit_web_p);
create index ph_photos_by_r_nonp_print_idx on ph_photos(rights_nonprofit_print_p);
create index ph_photos_by_r_comm_web_idx on ph_photos(rights_comm_web_p);
create index ph_photos_by_r_comm_print_idx on ph_photos(rights_comm_print_p);

create sequence ph_presentation_id_sequence start with 1 increment by 1;

create table ph_presentations (  
    presentation_id integer not null primary key,  
    user_id integer not null references users,  
    title varchar(200),  
    public_p char(1) check (public_p in ('t','f')),  
    beginning_note varchar(4000),  
    ending_note varchar(4000),  
    use_html_code_p char(1) check (use_html_code_p in ('t','f')),  
    html_code clob,  
    creation_date date
);

create index ph_presentation_by_user_idx on ph_presentations(user_id);

create table ph_presentation_photo_map (  
    presentation_id integer not null references ph_presentations,  
    photo_id integer not null references ph_photos,  
    annotation varchar(4000),  
    photo_order integer
);

create index ph_prest_photo_by_present_idx on ph_presentation_photo_map(presentation_id);
create index ph_prest_photo_by_photo_idx on ph_presentation_photo_map(photo_id);

create table ph_presentation_user_map (  
    presentation_id integer not null references ph_presentations,  
    user_id integer not null references users
);

create index ph_prest_user_by_present_idx on ph_presentation_user_map(presentation_id);
create index ph_prest_user_by_user_idx on ph_presentation_user_map(user_id);

-- Links to the General Comments module

insert into table_acs_properties  
(table_name, section_name, user_url_stub, admin_url_stub)
select 'ph_photos', 'photodb photos',
'/photodb/photo.tcl?photo_id=','/photodb/admin/photo.tcl?photo_id='
from dual
where 0 = (select count(*) from table_acs_properties where table_name = 'ph_photos');

insert into table_acs_properties
(table_name, section_name, user_url_stub, admin_url_stub)
select 'ph_presentations', 'photodb presentations',
'/photodb/presentation.tcl?presentation_id=','/photodb/admin/presentation.tcl?presentation_id='
from dual
where 0 = (select count(*) from table_acs_properties where table_name = 'ph_presentations');
B TCL and SQL Code for Results Data

# jkoontz@mit.edu Jan 2000
# Page to display Meng thesis data

ad_page_variables {
    {months 5}
    {times_per_month 4}
}

ReturnHeaders
ns_write "[ad_header "Meng Thesis Data"]"

<h2>Meng Thesis Data</h2>

[ad_context_bar_ws "Meng Thesis Data"]

<hr>

Months back: $months<br>
Points per month: $times_per_month<br>

""

set db [ns_db gethandle]

set query_list [list [list Users "select count ( distinct user_id ) from ph_photos where creation_date < "] [list Photos "select count ( distinct photo_id ) from ph_photos where creation_date < "] [list Cameras "select count( camera_id ) from ph_cameras where creation_date < "] [list Films "select count( film_id ) from ph_films where last_modified_date < "] ]

foreach query $query_list {
    ns_write "<h3>[lindex $query 0]<h3>\n"

    for {set time 0} {$time < $months} {incr time} {

        for {set point 0} {$point < $times_per_month} {incr point} {

            set upto_date "(add_months (sysdate, - $time) - ($point * 30 / $times_per_month))"

            set pretty_date [database_to_tcl_string $db "select to_char ( $upto_date , 'Mon DD') from dual"]

        }
    }
}
set sql_count [database_to_tcl_string $db "[lindex $query 1]$uptodate"]

ns_write "<B>$time</B>, $pretty_date, $sql_count<br>
#
# 4. each custom field user: percentage of photos w/ custom data
#(view each, are there any?)
} } 

ns_write "<h3>Custom Fields</h3>

set selection [ns_db select $db "select tochar(date added, 'Mon DD') as date_added, field_type, field_comment from ph_custom_photo_fields"]

while {{[ns_db getrow $db $selection]}} {
   set_variables after-query
   ns_write "$date_added,$field_type,$field-comment<br>
  }

ns_write 
<h3>Other Info</h3>

set unused_cameras [database_to_tcl_string $db "select count(*) from ph_cameras c where not exists (select 1 from ph_photos where camera_id = c.camera_id)"]

ns_write "
<ul>
<li>Number of Cameras without photos: $unused_cameras
</ul>
"

# Get the change in users/photos each day.

ns_write 
<h3>Number of users-photos/day</h3>
<ul>
<li>Date, number of people, number of photos
" 

set start_date "to_date('10-01-1999', 'MM-DD-YYYY')"
for {set i 0} {i < 200} {incr i} {
    set n_people [database_to_tcl_string $db "select count(distinct(user_id))
        from ph_photos where creation_date <= $start_date + $i"]

    set n_photos [database_to_tcl_string $db "select count(photoid)
        from ph_photos where creation_date <= $start_date + $i"]

    set pretty_date [database_to_tcl_string $db "select to_char($start_date + $i, 'Mon-DD-YYYY') from dual"]

    ns_write "<li>$pretty_date, $n_people, $n_photos
    "}

ns_write "
</ul>

[ad_footer]"
Bibliography

<http://www.aolserver.com/doc/>

<http://arsdigita.com/doc/>


<http://photo.net/hardware.html>


[Pho00a] Photo Database Introduction 2000  
<http://photo.net/photodb/photo-intro.html>

[Pho00b] Photo Database Project Homepage. 2000  
<http://jkoontz.arsdigita.com/photodb-dev/>

[Hyp00] HyperText Markup Language, 2000 <http://www.w3c.org/MarkUp/>