Project #: C-36-697
Center #: 10/24-6-R7708-0A0
Contract#: RK52994
Prime #: 
Subprojects ?: N
Main project #:

Project unit: COMPUTING
Project director(s): KOLODNER J L

Cost share #: 
Center shr #: 

Mod #: 
Rev #: 0
OCA file #: 
Work type: RES
Document: PO
Contract entity: GTRC

Active

Project unit: COMPUTING
Unit code: 02.010.300

Mod #: 

Rev #: 0
OCA file #: 
Work type: RES
Document: PO
Contract entity: GTRC

Active

OCA contact: William F. Brown 894-4820

Sponsor technical contact

ANDREW BENNETT
(404)494-3762

Sponsor issuing office

DEBBIE ROOKS
(404)494-8024

LOCKHEED AERONAUTICAL SYSTEMS CO.
86 SOUTH COBB DR.
MARIETTA, GA 30063

SPECIAL PROGRAMS PROCUREMENT
LOCKHEED AERONAUTICAL SYSTEMS CO.
86 SOUTH COBB DR.
MARIETTA, GA 30063

Security class (U,C,S,TS) : U
Defense priority rating : NA

Equipment title vests with: Sponsor X GIT

NONE PROPOSED

INITIATION OF C-36-697. FIRM FIXED PRICE PURCHASE ORDER. PER SPONSOR REQUEST,
INVOICE (DTD 12/23/92) WAS FAXED TO LASC FROM OCA/PID FOR FULL $20K.
GEORGIA INSTITUTE OF TECHNOLOGY
OFFICE OF CONTRACT ADMINISTRATION

NOTICE OF PROJECT CLOSEOUT

Closeout Notice Date 06/22/94

Project No. C-36-697_________________ Center No. 10/24-6-R7708-OA0_

Project Director KOLODNER J L_________ School/Lab COMPUTING____

Sponsor LOCKHEED AERONAUT SYS CO-GA/______________________________

Contract/Grant No. RK52994________________________ Contract Entity GTRC

Prime Contract No. ________________________________

Title TOOLS FOR CASE-BASED DESIGN ADDING SYSTEMS______________________________

Effective Completion Date 921231 (Performance) 921231 (Reports)

Closeout Actions Required: Date

<table>
<thead>
<tr>
<th>Action</th>
<th>Y/N</th>
<th>Submitted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Final Invoice or Copy of Final Invoice</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Final Report of Inventions and/or Subcontracts</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Government Property Inventory &amp; Related Certificate</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Classified Material Certificate</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Release and Assignment</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

Comments_______________________________________________________________________________

Subproject Under Main Project No. ______________

Continues Project No. ______________

Distribution Required:

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Y/N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Director</td>
<td>Y</td>
</tr>
<tr>
<td>Administrative Network Representative</td>
<td>Y</td>
</tr>
<tr>
<td>GTRI Accounting/Grants and Contracts</td>
<td>Y</td>
</tr>
<tr>
<td>Procurement/Supply Services</td>
<td>Y</td>
</tr>
<tr>
<td>Research Property Management</td>
<td>Y</td>
</tr>
<tr>
<td>Research Security Services</td>
<td>N</td>
</tr>
<tr>
<td>Reports Coordinator (OCA)</td>
<td>Y</td>
</tr>
<tr>
<td>GTRC</td>
<td>Y</td>
</tr>
<tr>
<td>Project File</td>
<td>Y</td>
</tr>
<tr>
<td>Other</td>
<td>N</td>
</tr>
</tbody>
</table>

_____________________________________________________________________________________


The Design-MUSE 1.0
User's Manual

A Shell for Case-Based Design Aids

Eric A. Domeshek
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
domeshek@cc.gatech.edu
Preface

About Design-MUSE

Design-MUSE is the Design Memory Utility for Significant Experiences. It is an experimental shell intended to ease construction of Case-Based Design Aids (CBDAs). CBDAs, in turn, are an experimental class of computer systems intended to aid designers by providing easy access to prior design experiences and the lessons that can be learned from those experiences. CBDAs are primarily aimed at aiding designers in the very earliest stages of very complex design tasks. We believe that conceptual design is a critically important and all-too-little studied or supported aspect of design.

Design-MUSE is a Macintosh application, written in Macintosh Common Lisp. It is one product of an ongoing research program at the Georgia Institute of Technology’s College of Computing the goals of which include a better understanding of conceptual design, and improvements in the efficiency and quality of the design process. Since this is continuing research, Design-MUSE is a constantly evolving system. In many cases, our ideas about what the system should be considerably outstrip our implementation and what is. Nonetheless, the system has reached a state where we can perform some initial tests in actual use. Such testing requires a stable and documented version of the system. Accordingly, this manual describes the use of Design-MUSE 1.0, a version frozen at year-end 1993.

About this manual

This manual is divided into six main chapters. The first chapter is a very brief survey of the major features of the Design-MUSE intended to provide just enough of a model of the system for intrepid Macintosh-savvy users to dive in and start learning on their own. The second chapter provides a broader overview of the system, including an introduction to CBDAs and comments on system requirements, intended users, overall structure, conventions and limitations. The next three chapters describe how to use the system in each of its three major modes: browsing, modifying, and defining. The “Defining” chapter is not yet complete. The final chapter on “Hacking” attempts to make up for the fact that there are many gaps in the current
implementation which require the user to go outside the system and work in Common Lisp to accomplish necessary tasks. This chapter too is incomplete.

Like the program it describes, this manual is a work in progress. As future releases fill in the many currently unimplemented features, the manual will have to be revised. As experience with the system leads us to reorganize or overhaul the user interaction and the interface, the manual will have to be substantially rewritten. The current contents of this manual however are intended to be an accurate description of Design-MUSE 1.0.

About Georgia Tech’s design cognition studies

Design cognition and the development of design tools, including CBDA technology, are active research topics in the AI group at the Georgia Institute of Technology’s College of Computing. Development of Design-MUSE and CBDAs is only part of the whole picture. Other work in the group focuses on process design, automated design problem solving, creativity, and design education.

The team building Design-MUSE and the particular CBDAs it has been used to construct is also studying cognitive models of design and developing theory and applications of AI techniques such as case-based reasoning (CBR). Design-MUSE is being used to develop educational systems. We expect also to use it as a base for exploring issues in computer supported collaboration in design.

CBDA prototypes built to date include Archie-II, a tool to aid architects with conceptual design of buildings, and MIDAS, a corporate memory for designers of aircraft subsystems. Initial analysis has been performed for a CBDA-like system to rationalize design of satellite ground control software. Experiments in using Design-MUSE to rapidly construct CBDAs in several engineering fields are now under way.

Acknowledgments

The work on CBDAs in general and Design-MUSE in particular has all been conducted under the direction of Professor Janet Kolodner in her AI laboratory at the Georgia Institute of Technology’s College of Computing. Initial work on CBDAs, culminating in a system called Archie, was performed in collaboration with Professor Ashok Goel also of the College of Computing, and with Professor Craig Zimring in the College of Architecture. Work with
Professor Zimring on Archie-II is ongoing, and Professor Goel continues to develop similar ideas in an independent set of projects, including AskJef, and ArchieTutor. This work has benefited greatly from all their contributions.

Design-MUSE and this manual owe much of their existence, however, to the Advanced Design Division of Lockheed Aeronautical Systems Company (LASC), and in particular, to the vision of Andrew Bennett and the efforts of Marcia Herndon. LASC funded the development of the MIDAS system used as an example throughout this manual, and Marcia was the first domain-expert/guinea-pig for our attempts at constructing a CBDA shell. Many aspects of the current system, not the least its relative lack of bugs, is due to Marcia's diligence.

The CBDA work has also involved a host of other research scientists and graduate students. Richard Billington was a member of the original Archie and Archie-II teams, as was Ali Malkawi. Terry Chandler has worked on a related set of projects, most notably SciEd. Anna Zacherl was a key contributor to Archie-II. Viji Narayanan, Ellen Do, Hussam Khalil, Osman Otaman, Ameen Farook, and Marin Simina all continue building Archie-II.

Finally, this work has been supported in part by the Defense Advanced Research Projects Agency, monitored by ONR under contract N00014-91-J-4092. Some of the Design-MUSE work was also supported by Lockheed Aeronautical Systems Company. All views expressed are those of the author.
Modifying class definitions
Chapter 1. Quick Start

Is the Quick Start for You?

This chapter is provided for those who want to dive right in and try using a CBDA. At the very least, you should be comfortable with the Macintosh and understand its basic conventions. The material will be most useful if you have the program running in front of you while you read through it. Refer to the program screens as features are introduced, and feel free to try out operations (menu choices, mouse actions, etc.) as they are described.

For those who are not ready to immerse themselves in the running program, this chapter may be too cryptic and confusing. However, after working through the rest of the manual, it may be worthwhile to come back to this section for a convenient summary of the program and its basic conventions.

In either case, we hope you enjoy exploring with your CBDA. We also hope you will share with us your impressions of the system and ideas for how it might be improved. You can send email to domeshk@cc.gatech.edu. Thank you for trying this out, and good luck.

Starting

- To start running your CBDA, double click on the icon representing the program. You may have to wait several minutes for the system to start up. You will know it has successfully started when a new window displaying a picture that looks like the cover of a spiral bound notebook appears on your screen.

- Before the initial Notebook screen appears, a dialog box may pop up to ask if you want the system to compile its data files. You may simply click “No”, and the program will continue loading, probably a good bit faster than if you click “Yes”.

- When the Notebook window appears, you should log in by filling in the two blank fields labeled “This book belongs to:” and “My password is:”, and
then clicking the "OK" button (or hitting the "Return" key). You can type into one of the blank fields by clicking the mouse inside of it; you can move between the two fields by hitting the "Tab" key.

- If your attempt to log in is successful, a small dialog box will pop up in the middle of the screen asking which window you would like to start with. You should probably choose Lessons (by clicking on that option) and then click on the "OK" button.

- If your attempt to log in is unsuccessful, an alert box will pop up to tell you so. Please correct your name and/or password and try logging in again.

- If you do not have an account, or do not know your log-in name and password, see whoever is administering your CBDA for assistance.

### Browsing

- A CBDA provides access to two types of information: 1) documentation describing existing designs, and 2) lessons that can be learned from those designs. Accordingly, the CBDA browsing interface has two major windows: the Designs window, and the Lessons window.

- Looking for information in a CBDA involves two main activities: 1) searching for designs or lessons, and 2) browsing among designs and lessons. To search, you construct a search cue, much as you would for a standard database system. To browse, you click on buttons or select menu items associated with a current display to make the system bring up information related to whatever is already being displayed. Both of these operations are described in much more detail below.

- Most windows in a CBDA are divided into several panes. Each pane takes up part of the window, and is dedicated to displaying some particular type of information. For instance, there are several types of lesson information: problems, responses, and stories. Accordingly, the Lessons window is divided into several panes, including one for problems, one for responses, and one for stories. These three types of lesson information and the connections among them are described in great detail below.

- The Designs and Lessons windows each include a pane for composing search cues. The Descriptions pane at the top of the Designs window allows you to tell the system what kinds of designs you would like it to retrieve from its library. The Interests pane at the top of the Lessons window allows you to specify the kinds of lessons you would like the system to retrieve. (Note that these panes are set off from the normal information display panes by a thick
black border). To compose search cues click on the empty blanks of the form displayed in the pane, and choose fillers from the pop-up menus that appear.

- Though a CBDA typically contains a large number of designs and lessons, each pane usually only contains some few selected items at a time, and generally only displays one of those selected items at a time. For instance, after a search on an Interests search cue, the Stories pane might contain a half dozen stories selected because they matched the cue; the highest ranking story would be the one that was actually displayed. Each pane has a header that tells you what kind of information it displays, and, when its selected set is not empty, it tells you how many items are selected and which one is currently displayed (so the Stories pane header might say “Story (1 of 6)”).

- Each pane’s header also contains pop-up menus. A menu called Choose allows you to control which of the items in the pane’s selected set actually gets displayed. A menu called Browse allows you to view other information related to the currently displayed item. Note that the Choose menu controls what is displayed in its own pane, while the Browse menu usually causes the contents of some other panes to change. For instance, the Browse menu in the Stories pane includes an item labeled “Problems”; choosing it causes the Problems pane to be assigned a new selected set containing only those problems linked to the currently displayed story. Also note that instead of a Browse menu, the two search cue panes (Descriptions and Interests) include a Search menu that controls when and how searching is done.

- Browsing also works between the two major windows. For instance, another item in the Story pane’s Browse menu is labeled “Designs”; choosing it causes the Designs window to come to the front, where it can display some documentation on the artifact the story is about. Similarly, it is possible to click on “hot buttons” in the design documents displayed in the Designs pane to bring the Lessons window to the front where it can display a story about the current design.

- The system tries to make sure that the information selected and displayed in its various panes at any given time is a coherent grouping. It bases its selections on whatever the user has most recently asked the system to display. For example, after an interests search selects a set of stories and causes the top-rated one to be displayed, the Problems and Responses panes will be loaded with sets of problems and responses relevant to that story. Similarly, after the “Responses” item from a Browse menu loads the Responses pane with a new set of responses, the Problems and Stories panes are loaded with new sets of problems and stories that relate to the new response on display. The pane containing the most recently selected item is called the focal pane; in the first example, it is the story pane, in the second, it is the response pane. The focal pane determines the contents of the other panes. You can tell which pane is the focal pane because it is framed in a lighter color than the other panes.
• In addition to browsing, is also always possible to move among a CBDA’s windows by selecting the name of the window you want to see from the **Windows** menu that is a constant part of the menu bar at the top of the screen. In addition to the **Designs** and **Lessons** windows, there is also a window for the initial log-in screen (called the **Notebook** window), and a window for displaying annotated bibliographic citations for all the other information in the system (the **Sources** window). Note that the **Browse** menu in most panes include an item labeled “Sources”, which causes the **Sources** window to come to the front where it displays citations for the pane’s current item.

• In addition to windows resizing in the normal way, you can also generally change the size of panes within windows so that they share the window’s screen space as you like. Panes are resized in one of two ways: 1) by clicking on special window-splitter widgets and dragging the division point to a new location, or 2) by choosing menu items from the **Layout** menu that is usually available in the menu bar at the top of the screen.

**Modifying**

• CBDAs are designed not just to support on-line browsing of design cases, but also to facilitate the construction of browsable libraries of information. Users may be assigned different levels of privileges. Users who have Modify privileges are allowed to change the contents of the case library.

• System modification is, for the most part, performed using the same interface as system browsing. The main difference is that users with modify privileges have access to one additional pop-up menu in each pane: the **Modify** menu. This menu provides commands for editing, creating, copying, and deleting whatever sort of information item is displayed in the pane.

• The first item in the **Modify** menu is labeled “Allow Editing”. Editing of the currently displayed item is only allowed when this menu item is checked. Selecting this item toggles whether or not it is checked, and thus whether or not editing is allowed on the current item. A CBDA remembers whether a particular item has editing allowed for the duration of a session.

• When editing is allowed, the mouse cursor turns from its normal arrow form into an I-beam when moving over the pane’s editable text fields. Text can be edited using normal Macintosh click, drag, and type conventions. Cut, copy and paste also work as usual.

• Many panes display items that have “invisible” data associated with them -- that is, data that is not obvious (and editable) when the item is displayed in its pane. For example, stories can have a one sentence summary that is not
displayed in the Stories pane, and they can be assigned outcome ratings such as *positive*, *negative*, or *mixed* which also are not reflected in the Stories pane. Choosing the “Story Fields” item from the Modify menu provides access to a dialog box where this invisible information can be specified for the current story. Similar menu items appear in the Modify menu in the Problems and Responses panes.

- Some panes display items that can be assigned indexes to be used as the basis for searching. For example, stories can be described as being relevant to a set of design interests. Choosing the “Story Interests” item from the Modify menu provides access to a new window where you can create interest forms (just like those that appear in the Interests pane) associated with the currently displayed story. A similar menu item appears in the Modify menu in the Problems pane.

- The Browse menu normally provides a way to traverse links between items. When you have Modify privileges, and when “Allow Editing” is checked, you may also create links between items using the Browse menu. The trick is to hold down the option key while choosing an item from the Browse menu. For example, with a problem on display in the Problems pane and editing allowed, you can option-click on the “Responses” item in the Problem pane’s Browse menu to create a link between the displayed problem and some response. A floating dialog box will appear at this point, providing options to cancel or create the link (as well as options to delete a link). Make sure the Responses pane is displaying the response that you wish to link to the original problem, then click “Create Link”. A link from the problem to the response will be created (as well as a link from the response back to the problem). In the future, choosing the “Responses” item from the Problem pane’s Browse menu when the problem is displayed will focus on a set of responses that includes the one you just linked.

- It is also easy to create links between graphics displayed in the Designs pane of the Designs window and stories displayed in the Stories pane of the Lessons window. Simply hold down the option key while clicking on a graphic, and the same floating link dialog will appear. Make sure your target story is on display in the Stories pane, and click on “Create Link” to establish the link between the graphic and the story. The link will appear as a small round dot on the graphic, positioned at the point you initially clicked. To move a dot to a new position, you may option-click on the dot, and then drag it to a new position.

- The Modify menu of most panes contains an item labeled something like “Select all <item type>“. This is useful when, in trying to create a link, you need to find the target for the link and arrange for it to be displayed in its pane. Use this in conjunction with the Choose menu to set up the link target.
• To delete a link that is no longer desired you should go through the initial action as if creating a new link, in order to get the floating link dialog to appear. Then you should select the undesired link from the scrolling list in that dialog by clicking on it. This will activate the dialog's "Delete Link" button. Click on it to delete the link.

• For any of the modifications you make to be included in the system's database, you must remember to save your work. You can either choose "Save" from the File menu in the menu bar at the top of the screen, or as you are quitting the system, you may respond "Yes" to the dialog asking if you want to save changes. Note that in the current system, saving data may take a few minutes, and the time grows as the amount of data increases.

**Defining**

• CBDAs are designed to make it easy to change some of the data structures used in case library. Some users may be assigned Define privileges, which allows them to make certain changes to the system that even users with Modify privileges cannot make. Users with Define privileges have access to all the facilities of a CBDA. Additional power is only available by dropping into the underlying Lisp system and monkeying with Lisp code. Note that in the current implementation the Lisp listener is always available, even to the lowliest browsing user. This should be corrected in later releases of the system.

• There is probably no way to get a quick start on defining a completely new CBDA. At present, a certain amount of Lisp coding is required to get a new system off the ground, including a bit of hand tailoring of data files. This also should be corrected in later releases of the system.

• There are, however, some facilities available with Define privileges that are important in building a CBDA. These are all provided through an entirely new window called the **Define** window. The three panes in the **Define** window allow display and editing of object classes, objects, and the lowest level groups of data items known as **spaces**.

• The Classes pane currently supports object class display, but not editing. Use this pane's **Choose** menu (particularly the item labeled "Pick One...") to view different class definitions. Note that the contents of the Classes pane determines the contents of the Objects pane.

• The Objects pane supports both display and editing of object instances. The selected set for this pane is all the instances of the class currently being
displayed in the Class pane. To edit a field of an object, click on the filler, and a dialog will pop up in which you can provide a new value.

• The Fillers pane supports display and editing of a set of data types peculiar to CBDAs. These objects are called *spaces*, and they are a way of defining primitive symbolic elements for a CBDA's representations. The primitives are called *names*, and each space has associated with it some set of names. Object classes have type constraints on their fields, and specifying a space as the constraint on a field means that the field's fillers must be names drawn from the space. Actually, these primitives are not completely primitive; names map to clusters of micro-features called *discrims*. You can think of a set of discrims defining the space, and the space’s names as picking out interesting combinations of discrims (or points in the feature space defined by the discrim set). Alternately, you can avoid thinking about this entirely...

• In any case, the Fillers pane displays a space as a grid, with names down the left margin, and discrims across the top. The definition of a name can be read off by looking across its row for those discrim columns that contain black squares. Patterns of similarity and difference between names can be read off by comparing the patterns of black and white squares. These definitions can be edited by clicking on squares to toggle between black and white. Names and discrims can be added using items in the pane’s *Modify* menu. Names and discrims can be deleted, renamed, or reordered, by clicking on their labels (at the heads of the rows or columns) and dragging them around.

• As with all modifications, you must explicitly save any changes you make if you want them to affect the system database. Saving data was described in the Modify section of this chapter.

**Quitting**

• To quit from a CBDA you chose the "*Quit*" item from the *File* menu in the menu bar at the top of the screen. Whether you have made any changes or not, the system will present a dialog asking if you want to save any changes you have made. If you have made any modifications to the system during the session, you should click "*Yes*". If you have not already saved your work by choosing "*Save*" from the *File* menu, then you may lose the changes you have made unless you click "*Yes*". If you have not made any changes that need saving, you can get out of the program more quickly by responding "*No*" to the query about saving the database.
Chapter 2. Overview

What is Design-MUSE?

This manual describes the Design-MUSE Case-Based Design Aid (CBDA) shell, and presents detailed discussion of system use. The purpose of a CBDA is to provide easy access to information about existing artifact designs to help designers working on new projects. The purpose of this shell is to make it easy to create CBDAs to assist many different kinds of designers. While much of the system is quite generic, its features and use will be illustrated primarily with examples from the MIDAS system, a prototype CBDA for initial design of aircraft utility subsystems built using Design-MUSE.

A CBDA is a tool to help a designer produce designs more successfully and more quickly. It is worth picking that goal statement apart and studying the pieces one at a time.

A CBDA is a tool... -- A CBDA is intended to be used by a person who retains primary responsibility for the work of creating a new design. It does not do design, it aids a user who is doing design.

...to help a designer... -- The person using a CBDA should be a designer engaged in a design problem. It is more important that the user be engaged in doing design than that the user be a fully competent expert designer. That is, a CBDA can be useful for student designers learning their craft, for experienced designers who want extra design expertise at their fingertips, or for part-time or amateur designers who need extra guidance.

...produce designs... -- The kinds of designs, or rather, the aspect of design, that CBDAs are intended to aid is what we call conceptual design. Conceptual design covers the early stages of a design project when the requirements are elaborated and schematic solutions are proposed. Thus, a CBDA is intended to help designers better understand the problems they are working on, including unstated requirements, interactions among constraints, and strategies that might lead to successful solutions.

...more successfully... -- We believe that use of a CBDA should result in better designs because the designer will have been helped to
consider more issues and more possible solutions. Exploration of the problem and of prior art are standard parts of most design efforts, but they are time consuming tasks, and often cannot be pursued as far as they ought to be.

...and more quickly -- Having information about previous designs and lessons from those designs easily available on line and organized for efficient browsing should make the exploration process not only more comprehensive, but also more efficient. Furthermore, a more complete job of research performed up front will usually pay dividends in the form of fewer costly revisions later on.

What is a CBDA?

CBDAs are intended to support conceptual design. Conceptual design is a dialog between problem specification and design proposals, where both are subject to continual revision. This makes conceptual design a highly recursive process, with the designer moving rapidly between high level "strategic" decisions and relatively detailed design decisions. To take an architectural example, a designer of a new courthouse may start out aiming for a long narrow building with services off of a central corridor, then after laying out a courtroom or other important area based on that initial commitment, may go back and refine the overall layout. Conceptual design requires fluidly mixing different types of thinking -- reasoning out the consequences of the initial problem statement by application of domain rules, design heuristics, and knowledge of prior cases. CBDAs aim to allow designers to move among these different approaches easily, and to take notes while doing so. Finally, conceptual design involves multiple modalities: words, sketches, diagrams, hard-line drawings, etc. CBDAs employ combinations of all these modalities, using vivid stories and clear graphics.

The very informality of conceptual design helps account for why CBDAs are useful. When problems are open-ended and there is no strong domain theory to support rule-based reasoning, reliance on cases is the standard way of coping. When we look to the real world we see design practitioners making significant use of previous cases as they work through the early stages of their problems. Prior experiences considered during conceptual design can suggest approaches to solving a problem or warn against proposed solutions that have been tried and found wanting. Just as likely, however, cases can serve primarily to elaborate the problem statement itself, illustrating sometimes non-obvious issues that deserve consideration, and offering ways of evaluating and critiquing partial solution proposals whatever their source.
Conceptual design should benefit from a quick, easy way to survey a wide range of existing artifacts. Currently such surveys are time consuming, costly and prone to omissions. The expense in time and money is due to the lack of well organized and easily accessible materials. To continue the architectural example, architects often flip through magazines and journals, but these are not particularly well indexed, nor do they contain much analytic material; less frequently, architects will travel around the world to visit relevant or landmark buildings as they research new commissions. With options like these, it is no wonder that the research process usually misses many relevant precedents; still less wonder that it will almost always miss interesting lessons that could be learned from less obviously relevant predecessors.

One way to think about a CBDA is as an on-line library of design experiences that saves the designer from having to trek to the library (or around the world), and that does a better job of collecting, organizing and presenting experience-based design lessons when they are relevant. CBDAs also combine raw documentation of existing designs with evaluations of the resulting artifacts, and carves those evaluative discussions up into small chunks focused to teach lessons about particular design issues and their interactions (yielding presentations we call stories). These stories are then indexed and cross linked with the design documentation, with design guidelines, and with other stories as well.

A CBDA is a clever hypermedia system. The cleverness lies in the kinds of information included, the ways that information is presented and the kinds of connections, or forms of browsing, supported. Our strategy is to understand the sorts of questions and issues designers should be considering, what they need to know in order to arrive at decisions, and how they prefer to visualize and organize the required knowledge. The result is a system combining documentation about existing designs with lessons learned from those designs, and organized so that each is accessible from the other.

Figure 2.1, for instance, shows a screen fragment from MIDAS, displaying a graphic of the hydraulic system from the A-7E jet. The light and dark colored dots on the schematic with lines leading off to annotations in the left margin are indicators of positive and negative stories associated with the hydraulic system of the A-7E. As one example of how a CBDA connects up information, clicking the mouse on the second annotation, “Using MS-6 resulted in increased component size and system weight” would lead to the display of the following story:

MS-6 is being evaluated as a non-flammable replacement for MIL-H-5606 and MIL-H-83282. When compared with these fluids, MS-6 has high viscosity, high density, and low bulk modulus. In an A-7E high pressure application study, MS-6 was substituted for MIL-H-5606 and tested for its performance.
Current hydraulic fluids break down above 6000 psi, so higher pressures require new fluid.

Using MS-6 resulted in increased component size and system weight.

Substituting an 8000 psi hydraulic system for a 3000 psi system yields volume reduction of 36%.

8000 psi hydraulic system has 40% lower probability of kill for flight controls than 3000 psi system.

Substituting an 8000 psi hydraulic system for a 3000 psi system yields weight reduction of 36%.

Components in wheel wells and engine bays were vulnerable parts of lightweight hydraulic system.

The system internal leakage was extremely low, but the fluid did have the tendency to foam and absorb air. This caused poor pump performance and resulted in a redesign to increase line diameters to maintain the system pressure drops. Since the total system volume increased, the reservoir size had to be increased as well. These redesigns for the MS-6 fluid resulted in a weight increase of 11.8% and a volume increase of 6.3%.

Stories like this do not stand on their own as isolated, uninterpreted experiences. A CBDA makes sure to connect them to other stories that touch...
on related topics. One way this is done is by linking stories to statements of
the problems they exemplify or address. The following is a problem
statement covering the MS-6 story:

While traditional hydraulic fluids provide effective power
distribution at a reasonable cost and weight, they tend to be highly
flammable. Hydraulic system fires have caused much loss of life and
introduced significant extra costs.

Story are also linked to statements describing possible general responses
to recurring problems. Here is a response to the problem above that also
relates to the MS-6 story:

Non-flammable or flame-resistant hydraulic fluids should be
used, where feasible, to reduce the chances of a system failure caused
by a fire. Non-flammable fluids currently under development are,
however, both heavy and costly. Overall system trades are necessary
to determine the applicability to each design case.

The point is that none of these forms of information is as effective
standing on its own as when appropriately linked to other presentations.
CBDA's provide a framework for useful information and for the connections
that enhance each independent item's value.

Given the right information and a set of meaningful connections, the
final piece required to make these systems work is a clever indexing scheme
that assures easy initial access to relevant materials. We assume users will be
engaged in working out a design, and so the emphasis in our indexing
scheme is associating stories and problems with the issues they address.
Indexes should allow users to describe their current commitments and
pending decisions as a basis for retrieving relevant advice.

System requirements

Design-MUSE runs on Macintosh computers. It is built using Macintosh
Common Lisp (MCL), which must be licensed separately.

Processor: A 68030 or better processor is strongly recommended.
Performance on Quadra class computers should be reasonable.

Memory: A 6 megabyte partition is recommended for starters. As
a CBDA's library grows, additional memory may be required to run
successfully.
Storage: The basic system takes up several megabytes on disk, and
that will grow as data is added to stock a CBDA's library. The
presentation media, in particular, are likely to eat up a lot of space.

Display: It is probably useless to try running on a "classic" Mac
with a tiny screen. This system can be run on a powerbook or standard
Mac II monitor, but really, you should have at least a full-page
monitor, and a two-page monitor is ideal. As it stands right now, the
system will not work on a black and white monitor; it requires at least
16-level gray-scale capability.

Who is this system for?

In creating and using a CBDA there are several distinct sorts of activities
that users might need to perform, and probably several distinct types of users.
Initially, to create a new CBDA, some users must define the data objects
peculiar to the new system. Each CBDA, for instance, is likely to deal with
different types of artifacts that have different descriptive features, and each
CBDA is likely to use a different ways of describing the lessons it has to teach
its users. Later, some users must be authorized to stock the CBDA's library by
adding new materials and modifying the relationships between existing
items. Finally, some users of this system will be end-users of a particular
CBDA empowered primarily to browse through the materials that have been
prepared by those responsible for defining and modifying the system’s
contents.

The shell, then, differentiates between three levels of privileges for users:
define, modify and browse. The user interface is designed to remain more or
less constant across these modes. Where certain windows, menus, or
commands are restricted to users with certain privileges, this will be noted in
the discussion.

System structure

Normal end-users of a CBDA are designers engaged in the conceptual
design of some new artifact. When using the CBDA, they browse through the
system’s library looking for design ideas, or for a better understanding of their
current problem. This assumption about who is using the system and what
they are using it for shapes the entire system design. The system makes use of
Figure 2.2. Schematic of Major CBDA Windows.

four major windows, each representing a major class of information or aspect of the system's function (see Figure 2.2).

The entry point to the system and the intended conceptual center or home base is the Notebook window. Since the point of using a CBDA is to advance ongoing design projects, the centerpiece of a CBDA is a personalized notebook in which designers record the lessons learned and design decisions made during their browsing. In addition to being the place for all active design work, the Notebook is intended to store interesting materials found during browsing, to maintain a history of the entire browsing session, and to provide access to on-line help.

The two large windows pictured beneath the Notebook window in Figure 2.2 organize most of the information in the system. The Designs window presents documentation about particular existing artifacts. The Lessons window
window presents design evaluations so users can learn the interesting lessons of those existing artifacts. Browsing in a CBDA means viewing information in these windows and moving among related presentations in order to follow up on interesting themes.

An important adjunct to both these windows, is the Sources window that allows all information in the system to be tied to citations identifying where the information came from. This is important not only to give credit to those who contribute their knowledge to the system, but also to allow users some way of deciding how much to trust the information presented to them. Much of the information in the system -- particularly the design evaluations -- is informed opinion; the (annotated) citations in the Sources window turn evaluations into something like signed editorials. The notion that there is rarely a single absolutely right answer when it comes to conceptual design of complex artifacts is an important idea that the CBDA shell tries to reinforce whenever possible.

Notebook window

The Notebook window is designed to look and act like a spiral bound notebook. It’s “cover” is the first image presented when a CBDA starts up; it provides a way for users to identify themselves to the system (as well as for the system to identify itself to users). In addition to its Cover, the Notebook can present five different faces to the user, corresponding to five different “sections,” each with a distinct purpose. Continuing the notebook metaphor, the sections can be selected by clicking on tabbed dividers along the right edge of the window (and also by selecting from a Sections menu that is part of the menu bar when the Notebook window is active).

The five sections of the Notebook are intended to manage the following kinds of information:

1. The Designs section is where users work on developing their new design. It functions most directly like a designer’s notebook.

2. The Lessons section is where users collect guidance from the system’s library. This part of the notebook lets users hold onto especially interesting items discovered while browsing.

3. The Notes section is where users keep any other notes they care to make while browsing.

4. The History section maintains a record of users’ activities as they work with the system. This should be an aid to maintaining orientation in the browsing “hyperspace.”

5. The Help section provide on-line instruction on system use.
In the current implementation, the Notebook window is primarily just a place holder for its intended functions. Only the Cover is fully implemented; the login feature sets the system’s mode according to a user’s assigned privileges. None of the five sections are yet implemented; in fact, only the Notes, History, and Help sections’ purposes are even well defined at this point.

**Designs window**

The Designs window displays documentation on existing designs. It is composed of two major panes (though the bottom pane is itself broken into two major parts).

1. The Descriptions pane displays a form that users can fill out to specify some features characterizing existing artifacts they might be interested in studying.

2. The Designs pane (composed of summary and display areas) presents documentation about existing artifacts retrieved from the system’s library. The pane is dominated by a large display area, but the left margin is reserved for short textual annotations highlighting interesting points about the displayed artifact. These annotations indicate the availability of stories that can be brought up in the Lessons window for more detailed study.

In the current implementation, the Designs window is missing some important features. Most notable is the lack of any way to control what view of an artifact will be presented (e.g., a textual description, a photo, plan, or schematic). Also lacking is a way for users with modify privileges to easily enter new artifacts and displays into the system’s library.

**Lessons window**

The Lessons window displays evaluations of existing designs along with general design guidance. It is composed of four major panes.

1. The Interests pane displays a form that users can fill out to specify a set of design issues and relevant artifact pieces that together characterize a set of lessons they might be interested in studying;

2. The Problems pane points out general negative outcomes that have been observed to impinge on the specified design issues;

3. The Responses pane collects general strategies that have been attempted in previous design to ameliorate recognized problems;
4. The **Stories** pane presents evaluated descriptions of particular designs that illustrate a problem, or that demonstrate a success or a failure in applying some response.

The **Lessons** window is fully implemented in the current system.

---

**Sources window**

The **Sources** window displays citations that identify where other information in the system came from. It is composed of two major panes.

1. The **Sources** pane contains a form that includes much of the information that would be included in a bibliographic citation;

2. The **Annotations** pane allows for arbitrary textual comments to be associated with a source.

The **Sources** window is fully implemented in the current system.

---

**General conventions**

This section briefly summarizes some very basic information about the way CBDAs are organized. It is often referenced in later discussions of specific parts of the system and how to accomplish particular tasks. It is worth familiarizing yourself with the concepts presented here, as they provide the structure for much of what is to come.

**Windows and panes**

As suggested in the last section’s discussion of system structure, a CBDA is organized to present its information in a set of **windows**, that are themselves each broken up in to a set of **panes**. Panes are the way the system organizes different kinds of information in the interface. Each pane specializes in displaying a given type of information. Figure 2.3, for instance, shows a screen shot of the Stories pane from the **Lessons** window. The Stories pane displays **stories** -- textual and graphical presentations reporting on particular experiences that can teach design lessons. Here a story about the C-141 aircraft entitled **"Avoiding Heat Damage to Control Wires"** is displayed along with an illustration of the aircraft’s wiring. Note how the pane’s header identifies the contents of the pane (e.g. **"Story (2 of 8)"**), Also note the presence of three standard pop-up menus in the header: **Choose,** **Browse,** and **Modify.**
During the C-141A all-electric actuation test, a wiring bundle experienced insulation damage and shorts. The problem was traced to the installation of the bundle in a high temperature area of the vehicle. The bundle was repaired and moved to a less severe environment. No further problems were experienced.

Figure 2.3. Screen Shot of Stories Pane from Lessons Window.

Figure 2.4. Vertical and Horizontal Splitter Icons.
Panes are grouped into windows to make it easier to view related information in an organized way. The size of each window is determined by the user, as for any Macintosh application. The panes within the window share the available space in a way that can be adjusted by the user at any time. To adjust the boundary between panes, the user can click the mouse on one of the “splitter” icons and drag it, with the mouse button down, until the division point is located in a more convenient place for viewing the current contents. The vertical and horizontal splitter icons are shown in figure 2.4. It is usually also possible to adjust pane size using the Layout menu described in the next section.

Menu bar menus

Since this is a Macintosh application, many of its commands are organized by the menu bar at the top of the screen. The menu bar always contains the standard Apple File and Edit menus. In addition, a custom menu called Windows lists the available major windows and offers immediate access to them. Selecting a window name from the Windows menu causes the corresponding window to be displayed and brought to the front for immediate use. The items in the Windows menu appear in an order reflecting how recently they have been used; the current window always appears as the first item in the menu (but that first item is never active, since it makes no sense to select the window that is already in front).

The currently active window determines what other menus (if any) appear in the menu bar. Usually, there is a menu called Layout that provides an easy way to adjust how the space within the active window is shared among the several panes from which it is composed. The first item in the Layout menu is always named “Balanced” and it simply divides the window space among the panes in some reasonable way; the balanced layout is the default, and the way that windows first appear. The remaining items in the Layout menu normally favor one pane at the expense of all the others, giving the chosen pane almost all of the space in the window (reserving only enough for the other panes so that they can still be seen, and potentially resized by dragging their splitter icons).

Standard pane menus

As noted in the discussion of figure 2.3, the header of each pane contains some pop-up menus for operations specific to that pane. There are three standard menus: Choose, Browse, and Modify. Note, however, that every pane does not offer all three of these menus, and, in fact, which menus are offered by any one pane can vary depending, for instance, and what privileges the user has. Here we will just briefly survey the organization and function of the Choose and Browse menus.
The Choose menu always contains the same first five items: "Pick One...", "First", "Previous", "Next", and "Last". All panes display one item at a time from some selected set of items. That is why the header of the Story pane says "Story (2 of 8)"; the pane currently has a selected set composed of eight stories and is displaying the second. Selecting "Pick One..." from the Story panes Choose menu at this point would bring up a dialog listing the titles of the eight current stories, and offering the chance to pick one for display. Selecting "First" would cause the pane to display the first of the eight stories. Selecting "Last" would cause it to display number eight of eight. "Previous" and "Next" would cause the pane to display stories one of eight, and three of eight, respectively.

The Browse menu appears in most, but not all, panes. Though it always serves roughly the same purpose, the options it provides differ from pane to pane, depending on what sorts of browsing make sense starting from the kind of information displayed in a particular pane. The first item on the Browse menu always offers the option of saving the pane's current item into the user's notebook (this option is not yet implemented; see "Caveats and limitations" below). Towards the end of the Browse menu, there is usually an item that offers the option of viewing the source records associated with the displayed item (which will bring up the Source window), and an item for viewing any notes associated with the displayed item (like the other notebook option, this too is unimplemented). Other options in the Browse menus of particular panes will be discussed throughout chapter 3.

Focal panes and input panes

The Stories, Problems, and Responses panes together display the contents of the system's lesson library (and, for users with modify privileges, can also be used to collect new lesson materials). At any given time, one of these three panes is taken to be the focal pane. The focal pane is the one displaying the last thing the user actually requested to see; this distinction matters because the contents of the non-focal panes is usually determined by the contents of the focal pane. If, for instance, the user has chosen to view a particular story, the Problems and Responses panes will be updated to display problems and responses that relate to the chosen story; likewise, the Designs pane will be updated to display some view of a design tied to the currently displayed story. To help the user keep track of what is being displayed, the focal pane is always highlighted using a border with a lighter shade than the other two panes.

Also note that the fourth pane in the Lessons window -- the Interests pane -- is different from the three just discussed in that it does not display or collect data from the system's library; it is solely for the user to express queries. The same is true of the Descriptions pane in the Designs window. These two input-only panes are set off from other panes in their windows by use of darker shades and thick black borders.
Caveats and limitations

As already noted, the current version of the Design-MUSE has several obvious major gaps. Some of these are described briefly here so that the user can have a better idea of what is intended, and of what is likely to be implemented in coming releases.

Notebook sections

Almost all features intended for the Notebook window are currently unimplemented. Only the Cover, with its login fields is complete. Choosing any section other than the Cover (either from the Sections menu that is part of the menu bar when the Notebook window is active, or from among the divider tabs along the right edge of the notebook) will cause the display of a blank notebook page with the message that the requested section is not yet implemented.

Saving and printing

Two of the major unimplemented features intended for the Notebook are worth special comment because they are so central to the entire system’s design and purpose. Throughout the browsing windows, menu options offer to save information into the Notebook; this feature is not yet active. The idea is that users should be accumulating a record of their research and decision making that they can later take away with them. Since the Notebook print facility is unimplemented, there is currently little point being able to save information into the Notebook. Expect these features to be among the first extensions to the current system.

Modifying design data

The system is designed to make it easy for those with modify privileges to enter new materials. In the current version, this ease of use works well for Lesson information, but breaks down when it comes to entering Design data. In particular, the Modify chapter documents the magic incantations currently required to enter new artifacts and their displays.

Defining object classes

The systems also aims to make it easy for those with define privileges to change the data structures and indexing for a CBDA. A fifth window (the Define window), not yet introduced, is provided for these purposes. In the
current version, however, the Classes pane of the Define window is severely limited. For now, expert users must fall back on the Lisp system text editor to make most changes to object class definitions.

Defining a new CBDA

In the current implementation, there is no easy way to create a completely new CBDA. Getting a new system off the ground requires a small amount of programming and mucking about in the underlying Lisp system.

Specific warnings

This section lists a number of specific issues and flaws that the user should be aware of when using version 1.0 of Design-MUSE

Saving data

Note that the system is fairly slow about saving out the database to disk. It may take several minutes to complete a save. Nonetheless, you are encouraged to take the time to save your work with some frequency. You are also encouraged to keep duplicates of the entire Data folder. Remember, this is still an experimental program, and glitches are always possible. Be careful, and maintain even more backups than you would with a commercial application. Remember: better “save” than sorry.

Unimplemented commands

Though attempts were made to clean up all the loose ends in the program for the current release, there are some unimplemented features that were not disabled or hidden away. Unimplemented yet enabled commands appearing in menus will usually result in a warning dialog appearing to inform the user that the requested feature is not available.

In some very few cases these dialogs were not provided so the underlying Lisp system generates an error. When this happens, a new small window titled Listener appears on the screen presenting an incomprehensible error message. Don’t panic. Feel free to ignore this window and the error message. Return to the window you were in and resume your work. Note that attempts to get rid of the Listener window will not really succeed, since closing the window will simply cause the system to create a new Listener. You might as well leave it alone since it is not doing any harm.
Frozen scrollbars

In some situations displays with scroll bars that should allow you to see all parts of items too large to fit in the available screen space will not respond in the usual Macintosh way to clicking and dragging actions. This is a minor bug that can be worked around by selecting the "Allow Editing" item from the Modify menu of the affected display pane, so that a check-mark appears next to the menu item. This situation only arises when the user has sufficient privileges to have access to the Modify menu.
Chapter 3. Browsing

Introduction

There are effectively four different ways to search and browse in a CBDA: *Interest-based search* makes use of the system's issue-centered indexes to retrieve lessons learned (in the form of problems, responses, and stories) starting from users' specifications of their current design concerns. *Problem-response-story browsing* allows users to move through the network of lesson components so that concrete examples are set in the context of design decisions, while abstract statements are illustrated by the specific examples. *Artifact-centered browsing* allows users to quickly scan lessons directly associated with existing artifacts they have chosen as particularly relevant models for their current design. *Description-based search* retrieves artifacts based on indexes that describe design features (rather than the design issues that compose the user interest specifications for lesson retrieval).

Interest-Based Search for Lessons

In interest-based search, users build a search cue using the menu-driven, fill-in-the-blank form in the Interests pane of the Lessons window. Then the CBDA finds lessons bearing on those interests. To support retrieval that supports designers' decision making, interests can include a specification of which parts of an artifact are being worked on and what issues are currently of concern. Significant lessons are usually the result of interactions among several issues, perhaps affecting several parts; they represent tradeoffs that past designers have made (knowingly or not). Interest specifications therefore allow the juxtaposition of multiple parts and issues.

There are several sorts of parts. In particular, for the physical artifacts we have so far considered, we focus on physical parts and functional parts. In building design, for example, physical parts translate to spaces (such as lobbies and courtrooms), while functional parts are systems (such as plumbing and electrical wiring). In aircraft subsystem design, we are mainly concerned with functional systems, such as the hydraulic system, and their component parts, such as pumps, lines, and fluid (see Figure 3.1, for example).

In other domains, dealing with different kinds of designs, we might pick out temporal or perhaps even social parts, as in a process or an organization.
By *issues*, we mean to include the broad range of things that designers must decide upon and that their decisions affect; this includes 1) explicit and implicit goals (the *requirements* or *intents* of the design), 2) the flip-side of such intents, properties to be avoided (the *pitfalls* that may yield an unacceptable design), and finally, 3) the specification of particular features, components or parameters as part of the artifact design.

In our current systems, we have attempted to keep the definition of issues broad by explicitly including in the interests specification a way of indicating the relevance of multiple stakeholder viewpoints and the different phases of the designed artifact's life cycle. This reinforces notions now gaining popularity in methodologies such as concurrent engineering, and helps account for the involvement of companies like Lockheed with this form of design aid. Figure 3.1 presents a sample interest for the current MIDAS system.

A search is initiated by choosing the "Search" option from the "Browse" menu of the Interests pane. That same menu provides a series of option settings controlling exactly how search will be done: what kind of information will be retrieved (problems versus stories), what range of information will be considered (all items in the library versus various subsets of the available information), and what basis will be used for judging matches between the user's current specified interests and the descriptions assigned to the target items (including options such as most overlap, any overlap, and exact match). The result of a search is a selected set of problems or stories, with the best matching item immediately displayed as the focus of attention (and with related information displayed in the other panes).

<table>
<thead>
<tr>
<th>System</th>
<th>Hydraulic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>Fluid</td>
</tr>
<tr>
<td>Issue</td>
<td>Safety</td>
</tr>
<tr>
<td>Life cycle</td>
<td>Use</td>
</tr>
<tr>
<td>Stakeholder</td>
<td>Pilot</td>
</tr>
</tbody>
</table>

**Figure 3.1. A Sample Index from the MIDAS System.**

**Problem-Response-Story Browsing**

The second browsing mechanism assumes that some problems, responses, or stories have already been selected (possibly through the interest-based search just described). Browsing proceeds from that point through the network of links among these presentations. The idea is that the network of problems, responses and stories should together provide specific and
memorable advice, while also setting that advice in a more general context. The general context, in turn, is made clearer by the set of specific stories that flesh out what might otherwise be vague abstractions.

A story describes some aspects of a design and recounts some consequences for the resulting artifact with respect to some of its associated stakeholders. The story in Figure 2.3, for instance, recounts one failure that occurred in routing electric control lines. If that story is of interest to a designer, then chances are there are other stories in the system that would also be of interest -- stories that have to do with other problems encountered in routing such lines, stories about successful instances of routing such lines,
and so on. There are also, most likely, problem and response statements in the system that will be relevant to the designer's interests. For instance, a relevant problem describes the interaction between efficiency of wire runs and safety of the resulting system; that problem in turn provides access to other stories, such as a situation in a test version of the C-141A where control wires picked up electrical interference and the flight surfaces made movements that the pilot had not commanded. Figure 3.2 shows the Problems, Responses, and Stories panes displaying this constellation of texts.

The lessons network is traversed by selecting options from the “Browse” menu, included in the header of most panes (see, for example, Figure 3.2). In the Stories pane, that menu provides ways to see related problems and responses (as well as design graphics and source citations). Browse menu commands gather the linked items and assign them as the new selected sets in those items’ panes. From the Responses pane, for instance, one option selects a set of stories that illustrate successes in implementing the response, while another selects those stories that illustrate failures in implementing the response. In both cases, browsing from the Responses pane leads to selection of new stories in the Stories pane. From there, of course, it is possible to browse on to other problems or responses those same stories might illustrate.

Artifact-Centered Browsing

Interest-centered search and browsing through the network of linked problems, responses and stories makes sense for designers who have become engaged in their design task to the point where they have identified a set of concerns. Before a designer can state a coherent set of interests, however, it is often useful to browse a set of existing artifacts to get an idea of the issues that ought to be in play. While a designer is browsing based on interests, it is sometimes useful to organize a large set of relevant stories in terms of the artifacts they bear on. In addition, after a designer has identified and studied a rich set of issues that have mattered in the past, it may become useful to return to the bigger picture and examine the solutions of the past without pre-straining them through the sieve of a specified set of issues. CBDAs provide for artifact-centered browsing to support all of these needs.

A user may start from a particular existing design, and the system will present documentation in the Design window accompanied by indications that there are relevant stories. In the prototypical case, the documentation is a floor plan for a building, or a schematic for an aircraft system, and the story indicator is a small dot placed next to an implicated part of the diagram. Selecting a set of dots causes their annotations to be displayed along-side the graphic as in Figure 2.1 (and causes the associated stories to be loaded as the selected set in the Stories pane). The annotations offer one sentence summaries of the associated stories. Clicking on an annotation takes the user to the Lessons window, where the associated story is displayed in the Story
pane. This then provides an entrée to the problems-responses-stories network. Since it is always possible to copy the description of a displayed story to the Interests pane, it also offers a way to begin a new interest-based search.

When a search results in the retrieval of many stories, the user can browse through the set of retrieved design documents corresponding to those stories. As in Figure 2.1, which shows the results of a search for stories discussing the interaction of hydraulic system pressure and weight, the annotations for the retrieved stories are already displayed alongside the design documentation. The short summaries now provide a quick way to survey and organize the results of the search. As before, the user is free to select any of the other visible dots to shift from exploring the original interests to browsing for other interesting issues.

Finally, browsing through the documentation for an existing design can be informative even without constant detours to explore associated stories. This is particularly true after a significant amount of exploration has already sensitized the designer to the relevant issues. The designer should then be able to do much of his or her own critiquing, and should be on the lookout for solution features that address those issues that are most salient in the current situation.

**Description-Based Search for Artifacts**

In many cases, the most likely starting point for a CBDA browsing session is a set of precedent designs. What we want is something similar to interest-based search, but for retrieving artifacts rather than for retrieving lessons. This mechanism is not yet fully implemented, but the Descriptions pane of the Designs window is intended to play a role much like the Interests pane of the Lessons window -- the user will fill in a partial specification of an artifact, and the CBDA will search for similar artifacts.

This final form of browsing resembles behavior supported by a tool being built at Lockheed; that system has as its main purpose the identification of the most similar existing aircraft to a new concept under development. Based on features such as mission, range, payload, etc., the tool tries to retrieve documentation on the most relevant aircraft in its database to support sizing estimates for utility subsystems. This kind of gross artifact-level matching is often a reasonable way to get started on a new project. Our claim, however, is that it is only a way to start, not a full answer in itself. In our CBDAs, we see much of the utility of this type of search deriving from the resulting access to more specific stories, with the concomitant raising of issues, highlighting of interactions, and proposal and illustration of responses.
How to...

This section is devoted to describing how to carry out most of the specific operations a user would want to perform while browsing in a CBDA. This section does not provide any sort of overview that would explain why you might want to be browsing, or exactly what you might be browsing for. That sort of perspective is provided by the “Overview” chapter of this manual and by the “Introduction” section of this chapter.

Searching on interests

One common way of getting started on finding useful information in a CBDA is to search through the library of problems or stories using a query specified in the Interests pane of the Lessons window. This section describes the procedure you would go through to perform this kind of search. There are three main steps: specifying interests, setting search options, and running the actual search. One final topic covered in this section is managing a set of search interests accumulated during a session with a CBDA.

Specifying interests

To specify interests for searching the first step is to make sure there is an interests form on display in the Interests pane. When the system starts up, a single blank interests form is provided in that pane. Figure 3.3 shows what the Interests pane looks like when the MIDAS CBDA has just started up. The precise slots of the interests form will vary from system to system; in MIDAS, at the time of this writing, interests have five slots called system, component, issue, lifecycle and stakeholder.

To fill out the form you can use the following techniques:

To specify a filler for a slot: Click the mouse on an empty box in the right hand column (empty, that is, except for the black downward-pointing triangle that indicates the box will behave like a pop-up menu). While holding the mouse down, move it over the menu that appeared when you clicked until it is over the filler that you want to choose for the slot. At that point, release the mouse button, and the chosen filler will be stored into the slot.

To replace a filler in a slot: Choose a filler from the pop-up menu of a filled slot just as if the slot were empty. Your new choice will replace the old filler.

To add additional fillers to a slot: Choose a new filler for a slot, just as if you were going to replace the old filler, but hold down the shift key while you click and release the mouse. Note that this only
works for those slots that are allowed to have more than one filler. If the slot is allowed multiple fillers, then the form will expand to include multiple lines labeled with the same slot name. If the slot is only allowed a single filler, then holding down the shift key while selecting a filler will simply replace any prior filler.

To delete a filler from a slot: Click the mouse and release it while over the name of the slot you want to empty. If a slot has multiple fillers, then clicking on one copy of the slot’s name only erases the filler on that line.

Setting search options

There are currently two sets of options available for controlling how interests search is performed in a CBDA. These options are controlled by a series of check-mark items in the Search menu of the Interests pane. Select one of the items in a set and it to become the current choice for that option; it appears in the menu with a check-mark next to its name, and the previously selected item loses its check-mark. The first set of options controls what kind of items are retrieved: the choices are “For Problems” or “For Stories”. The second set controls the criteria used for retrieval and has four options:

"Matching Most of Interests": This option causes the system to do a kind of “nearest neighbor” matching. It computes a weighted overlap between your specified interests and the labels attached to items in memory. Those items that score above a threshold will be included in the retrieved set.

"Matching Any of Interests": This option will retrieve an item with a label that contains any of the features specified in your interests.
"Matching All of Interests": This option will only retrieve an item with a label that contains all of the features specified in your interests.

"Matching Exactly to Interests": This option will only retrieve an item with a label that contains all the features specified in your interests, and only those features.

### Running searches

Actually requesting that the system perform a search through its memory is simple, once a search cue has been specified and the search options selected. Choosing the first item from the Interests pane’s Search menu, itself called “Search” will start the CBDA working. How long the system takes to retrieve the requested items depends on the size of the library and the complexity of the search cue. When the search completes, the system will either put up a dialog notifying you that it could not find anything that matched sufficiently well, or it will load the retrieved items into the appropriate pane (either the Problems pane or the Stories pane), make that pane the focal pane, and update other panes accordingly. See the first chapter’s section “General conventions” for a discussion of the notion of the focal pane.

### Managing interests

A CBDA starts up with a single blank interests form in the Interests pane. Additional blank forms can be created by selected New from the Choose menu of the Interests pane. Forms can also be created by selecting Copy from the Choose menu of the Interests pane (but of course interest forms created that way may not start out blank). The only way you might get into a situation where there is no interest form in the Interests pane is if you have deleted all the existing interest forms. The currently displayed index form can be deleted by selecting Delete from the Choose menu.

The first five options in the Choose menu of the Interests pane are the same as in all Choose menus throughout the system. They allow you to choose which of the accumulated interest forms will be displayed in the pane and used for searches. These commands were described in the “Overview” chapter under the heading of “General conventions”.

### Browsing from stories

Whenever there is a story on display in the Stories pane (whether as a result of an interests search, or of some other type of browsing) it is possible to browse for additional information related to that story. This section details all the options for moving to related information starting from a story. These options are all available as items on the Choose and Browse menus in the header of the Story pane. These menus were introduced in the “General conventions” section of chapter 2.
From a story on display in the Stories pane, the Browse menu can take you to related problems, responses, and designs (this is in addition to the basic Browse options discussed in chapter 2). Note, however, that menu items are only active when they make sense; for instance, the “Problems” item will be disabled when a story has no associated problems. When related items are available, browsing loads them into the relevant panes. If there is a single related item, it is loaded immediately; if there are several related items, a dialog box pops up first offering a choice of which item to display initially. In either case, the related item(s) become the new selected set for the target pane. Browsing to related problems or responses will also cause the Problem or Response pane to become the focal pane.

Moving among selected stories

The Stories pane, like all other panes, has a Choose menu that allows you to move among the currently selected set of stories. The options on the Choose menu were discussed in detail in the “Overview” chapter in the section on “General conventions”.

Finding related problems

The second item on the Story pane’s Browse menu is labeled “Problems”. Selecting this command causes the problems associated with the currently displayed story to be loaded into the Problems pane which becomes the focal pane. If there is more than one problem linked to the story, a chooser dialog will appear asking you to select which of the problems should be displayed first. This menu item is disabled if the current story has no linked problems.

Finding related responses

The third item on the Story pane’s Browse menu is labeled “Responses”. Selecting this command causes the responses associated with the currently displayed story to be loaded into the Responses pane which becomes the focal pane. If there is more than one response linked to the story, a chooser dialog will appear asking you to select which of the responses should be displayed first. This menu item is disabled if the current story has no linked responses.

Viewing associated designs

The fourth item on the Story pane’s Browse menu is labeled “Designs”. Selecting this command brings up the Designs window; note that this window was already loaded with the designs associated with the current story as soon as that story was displayed. If you have enough room on your screen you can always see the documentation on the designs that provide context for displayed stories at the same time as you see the stories. This menu item is disabled if the current story has no linked designs.
Checking the story source

The fifth item on the Story pane’s Browse menu is labeled “Sources” and is one of the standard options on all Browse menus. Selecting this command causes the sources associated with the currently displayed story to be loaded into the Sources window which is then made active. If there is more than one source linked to the story, a chooser dialog will appear asking you to select which of the sources should be displayed first. This menu item is disabled if there are no sources linked to the currently displayed story.

Copying story interests

The last item on the Story pane’s Browse menu is labeled “Copy Interests”. Selecting this command gathers all of the labels assigned to the currently displayed story, makes copies of them, and adds those copies to the set of interest forms available in the Interests pane. This is a useful option if you find an interesting story and want to search through memory for more stories that are like it. Once the duplicate interest forms are loaded into the Interests pane, you may use the Choose menu to display them, edit them as you would any interest form, and use them for further searches.

Browsing from problems

Browsing from problems is much like browsing from stories. Whenever there is a problem on display in the Problems pane (whether as a result of an interests search, or of some other type of browsing) it is possible to browse for additional information related to that problem. The Browse menu can take you to related problems, responses, and stories (this is in addition to the basic browse options discussed in chapter 2). When related items are available, browsing loads them into the relevant panes. If there is a single related item, it is loaded immediately; if there are several related items, a dialog box pops up first offering a choice of which item to display initially. In either case, the related item(s) become the new selected set for the target pane. Browsing to related problems, responses, or stories will also cause the Problem, Response, or Story pane to become the focal pane. The only novel parts of this Browse menu are the items that provide access to other problems.

How problems are related to other problems

There are three items on the Problems pane’s Browse menu giving access to related problems. The first, labeled “Related Problems” loads any problems that are linked to the current problem as the new selected set of problems; this includes other problems that are linked as either more general or more specific problems, plus any that are related in some unspecified way. The second item, labeled “More General Problems” loads only those problems that bear on the same issues as the current problem, but stated in a more
generic way. The third item, labeled “More Specific Problems” loads only those problems that bear on the same issues as the current problem, but stated in a more detailed way.

**Browsing from responses**

Browsing from responses is very much like browsing from problems. Whenever there is a response on display in the Responses pane it is possible to browse for additional information related to that response. The Browse menu can take you to related problems, responses, and stories (this is in addition to the basic Browse options discussed in chapter 2). When related items are available, browsing loads them into the relevant panes. If there is a single related item, it is loaded immediately; if there are several related items, a dialog box pops up first offering a choice of which item to display initially. In either case, the related item(s) become the new selected set for the target pane. Browsing to related problems, responses, or stories will also cause the Problem, Response, or Story pane to become the focal pane. The only parts of this Browse menu that need separate comment are the items that provide access to other responses and to stories.

**How responses are related to other responses**

There are two items on the Responses pane’s Browse menu giving access to related responses. The first item, labeled “More General Responses” loads only those responses that address the same problem using the same basic strategy, but that are stated in a more generic way. The second item, labeled “More Specific Responses” loads only those responses that bear on the same problem, and that uses more specific version of the same strategy.

**How stories are related to responses**

There are three items on the Responses pane’s Browse menu giving access to related stories. The first, labeled “Success Stories” loads stories linked to the current response as successes; these will usually be stories that have been labeled with a positive outcome. The second, labeled “Failure Stories” loads stories linked to the current response as failures; these will usually be stories that have been labeled with a negative outcome. The third, labeled “Mixed Outcome Stories” loads stories linked to the current response as mixed outcomes; these will usually be stories that have been labeled with a mixed outcome. (For discussion of outcome labeling of stories see “A tour of the designs pane” in the next section).

**Browsing from designs**

Whenever there is a design on display in the Designs pane (whether as a result of an descriptions search, or of displaying a story) it is possible to browse
for additional information related to that design. This section details all the options for moving to related information starting from a design. Some of these options are available as items on the Choose and Browse menus in the header of the Design pane. In addition, pieces of the displays in the Design pane also function as buttons that provide access to related information. Figure 2.1 presented a screen shot of the Designs pane from MIDAS showing a typical design graphic.

A tour of the designs pane

The bulk of the Designs pane is taken up by an area for displaying design graphics. In figure 2.1, this is the large scrolling area in the bottom right of the pane showing the hydraulic system schematic from the A-7E. The two labels fields above the graphic tell you that the artifact being displayed is the A-7E, and that the part you are currently viewing is the A-7E Hydraulic system. The column to the left of the graphic displays one sentence summaries of stories associated with the A7-E Hydraulic System; lines from those summaries run to dots distributed over the graphic, giving an indication of what part of the display is associated with each story. Note that the dots are color-coded to give a quick visual hint about the type of story: black dots indicate that the story is about something that went wrong with this plane; white dots indicate that the story tells about a particularly successful aspect of the design; gray dots indicate that the story tells of a mixed aspect of the design.

Selecting other designs

There are three ways to select other designs for display in the Designs pane. First of all, if there is more than one design in the pane’s selected set, then the standard Choose menu options will work in the standard way. Secondly, the Browse menu in the Designs pane contains two options that allow you to select other designs: the “ Constituent Parts” menu option gathers the items from the system’s library that are parts of the current item (in MIDAS, systems are parts of aircraft); the “ Containing Artifact” menu option gathers the items from the system’s library of which the current item is a part (in MIDAS, aircraft contain systems). Finally, the left label field (the field that in figure 2.1 says “A7-E”) also functions as a mouseable button; when you click on it, you get a dialog offering you a choice of all the top-level artifacts in the system’s library, and choosing one causes it to be displayed in the Designs pane.

Selecting stories

The Designs pane also offers way of browsing to stories, even though no story options appear in the pane’s Browse menu. The story dots appearing on the design graphic and the story summaries in the left margin are all mouse sensitive, providing a variety of ways to select and display stories.
To display a selected story: Stories that are linked to the currently displayed design (that is, stories that appear as a dot on the design graphic), and that are already part of the selected set for the Stories pane will have their summaries displayed in left margin of the Design pane (if there is enough room in the column). Clicking on a summary with the mouse will bring the Lessons window to the front and cause the corresponding story to be displayed. Similarly, clicking on the dot, instead of the summary will have the same effect.

To display any story: Any story linked to the currently displayed design can be selected by clicking the mouse on its dot, which will cause the Lessons window to come to the front with the Stories pane displaying the chosen story. If the story was not already part of the selected set, however, clicking on it causes that story to become the sole member of a new selected set in the Stories pane.

To add a story to the selected set: Clicking on the dot for a currently unselected story dot while holding down the shift key will not cause the system to make that story the only member of a new selected set. Instead, it will add that story to the current selected set. When adding a story in this way, the Lessons window is not brought to the front, and the currently displayed story does not change.

To remove a story from the selected set: Clicking on the dot for a currently selected story dot while holding down the shift key will cause the system to remove that story from the selected set. When removing a story in this way, the Lessons window is not brought to the front, and the currently displayed story only changes if the removed story was the one currently being displayed.

To select a new set of stories: Clicking somewhere in the design graphic, holding the mouse button down, and dragging the mouse to another location, will temporarily draw a rectangle on the graphic; the stories associated with each of the dots within that rectangle will be gathered to form the new selected set in the Stories pane. The Designs window remains the active window.

To add a set of stories to the selected set: Clicking and dragging a rectangle as in the previous procedure, but while holding down the shift key will cause the stories associated with the dots inside the rectangle to be added to the selected set of stories in the Stories pane.

---

Searching on descriptions

Searching for existing designs based on artifact descriptions works much like searching for lessons based on interest descriptions. It is another good
way to get started on finding useful information in a CBDA. This section describes the procedure you would go through to perform this kind of search. Just as with interest-based search, there are three main steps: specifying a description, setting search options, and running the actual search. Managing the accumulated set of search descriptions is also just like managing interests in the Interests pane. Please refer to the section on “Searching on interests” earlier in this chapter.

Specifying descriptions

Where figure 3.3 in that section showed a sample blank interests form, here figure 3.4 shows what the Descriptions pane looks like when the MIDAS CBDA has just started up. The precise slots of the description form will vary from system to system; in MIDAS, at the time of this writing, a description has three slots called toqw, customer, and mission. For details on how to fill out the descriptions form, see the discussion of “Searching on interests”.

Setting search options

These is currently only one set of search options applicable to controlling how descriptions search is performed in a CBDA. This set of options controls search criteria, and offers the same choices as are available for setting the criteria of interests search. Please see the section “Searching on interests” for details on those options.

Running searches

To search the system’s memory for design similar to the description you have prepared, simply select the first item from the Descriptions pane’s Search menu; that item is itself labeled “Search”. How long the system takes to retrieve the requested items depends on the size of the library and the complexity of the search cue. When the search completes, the system will either put up a dialog saying that it could not find anything that matched sufficiently well, or it will load the retrieved items into the Designs pane.
Figure 3.5. The Sources Window.

Viewing sources

The Sources window is always available using the "Sources" item in the Windows menu on the menu bar. However, when browsing, in order to see anything useful in the window, it is usually necessary to select the "sources" item from the Browse menu of some pane displaying an item that has been assigned a source or some sources. Figure 3.5 shows the Source window when it is empty.

Moving among sources

When more than one source has been selected, then you may choose among the selected set of sources by using the Choose menu of the Sources pane just as you would in any other pane. See the section on "General conventions" for details.

Moving among annotations

Each source item, like a bibliographic citation, can have associated with it a set of comments or annotations. These annotations might include
additional information about the source that would not fit into the standard citation format, or they might include any other comments that a builder of the system thought was important for users to know about the source. When you choose to display a source in the Source pane, any linked annotations are loaded into the Annotations pane. You can then choose among the selected annotations using the Choose menu, just as you would choose among any selected set of items.
Chapter 4. Modifying

Introduction

Design-MUSE is a major improvement over earlier ways of building CBDAs because it lets domain experts to handle many aspects of constructing and maintaining the system. A corporate memory stands a much better chance of catching on and of being sustainable if it allows its users to control what goes in, and if it does not impose high administrative costs. System-building chores are split into two sets: modifying, and defining. This chapter covers modifying, which includes the simple tasks that could be mapped most directly into the normal browsing interface. This includes adding new problems, responses, stories, and sources (or editing existing ones), indexing the problems and stories (or editing existing indexes), and creating links among the CBDA presentations (or changing existing links). A major gap in the current system is that modify mode does not provide any friendly interface for creating or editing design information.

In some future version of this manual, this section ought to contain explanations that make clear what kinds of information should be packaged into each of the available CBDA presentation types, and should offer guidelines for preparing effective presentations. These “Style sheets” are still under development as part of the ongoing CBDA research program. Instead, this section will give a quick once-over the various modification commands. This probably duplicates the material in the “Quick Start” chapter section on “Modifying”.

The major difference between the system as it presents itself to a browsing user and the system as it presents itself to a user with modify privileges is the addition of a new Modify menu to most panes (this menu is visible, for example, in figure 2.3). In addition, where Browse menu items normally traverse existing links (picking out new selected sets for other panes), modify mode lets the user initiate the creation of new links simply by holding down the option key while selecting an item from the Modify menu.

A pane’s Modify menu typically offers four major sets of commands: 1) toggling whether or not the currently displayed item can be edited; 2) creating, copying, and deleting items; 3) selecting all available items for the pane; 4) inspecting and specifying invisible background information about the currently displayed item. The first item is always labeled “Allow Editing”; it is
displayed with a check mark when editing of the current item is allowed, and selecting this item toggles whether or not the check mark is present. The exact labels of the next three items vary in each pane; in the Story pane for instance, they are “New Story”, “Copy Story”, and “Delete Story”. Adding a new story to the system starts with selecting the “New Story” item, which creates and displays a blank story. For new items, “Allow Editing” is checked automatically, so the user can simply type directly into the title and text fields. Text editing works as for all Macintosh applications; pointing and clicking or dragging, typing, tabbing, and cut, copy and paste all work as usual.

There is, however, more to a story than what is displayed in the Story pane. The last two items on the Story pane Modify menu are “Story Fields...” and “Story Interests...”. Choosing “Story Fields...” brings up a dialog box that allows the user to write a summary for the story (which can appear as an annotation in the Designs window when the story is selected), provides a field for specifying a PICT file (or soon, a QuickTime movie file) to accompany the story text, allows the user to rate the outcome of the story as positive, negative or mixed, and to describe the story as being about a single point or about an issue interaction. Choosing “Story Interests...” brings up a new window with a single pane identical to the Interests pane of the Lessons window. This window allows the user to assign a label to the story that will be used in matching user interests in the future. Any number of these indexes can be assigned to a story (and likewise, to a problem).

Having created the story, provided a title and text, filled in its background fields, and assigned indexes, the remaining task is to link it to other information items in the system, such as problems, responses, and design displays. This is done by selecting the desired kind of link from the Browse menu while holding down the option key, maneuvering in the target pane to the desired item, and clicking “Create Link” on the floating dialog that appeared when the Browse menu item was chosen. A similar mechanism creates links that appear as dots on design graphics. The linking dialog can be used to delete existing links as well as to create new ones.

How to...

This section is devoted to describing how to carry out most of the specific operations a user would want to perform while modifying a CBDA. This section does not provide any sort of overview that would explain what modifications make sense. That sort of overview is provided by the “Overview” chapter of this manual and by the “Introduction” section of this chapter.
Modifying stories

As one of the central presentation forms in a CBDA, stories are good representatives of how modification works in Design-MUSE. This section describes in detail all the modification operations available for stories.

Adding a new story

To add a new story to a CBDA, go through the following steps:

Select "New Story" from the Story pane’s Modify menu: A new blank story will be added to the selected set in the Story pane (and to the system’s library of stories). The story will have a meaningless computer-generated title but no text (and no specified project). Editing is automatically allowed on this new story.

Click in the Story pane’s projects field: A dialog will pop up listing all the projects that the system currently knows of. Click on the project that the new story is about, then click “OK”. You can also double click on a project as a shortcut to bypass having to click “OK”. If you want to associate more than one project with your new story, you can select several projects in the dialog by holding down the command key while clicking on projects. If you decide you don’t want to modify the contents of the projects field, you can click on the “Cancel” button of the projects choice dialog.

Type a title in the Story pane’s title field: Use standard Macintosh text editing techniques to enter the title of the new story in its title field. The text cursor should be in the title field, but you will probably want to drag over the bogus automatically generated title so your new typing will replace the old contents of the field.

Enter the text in the Story pane’s text field: Again, use standard Macintosh text editing techniques to enter the text of the new story. Make sure the text cursor is in the text field by clicking inside the field, or by hitting the tab key to move from the title field to the text field. If you have already prepared your story text in advance using a separate word processing program, you can easily transfer the text into the CBDA using copy and paste commands from the File menu.

Select “Story Fields...” from the Story pane’s Modify menu: A modal dialog box like that pictured in figure 4.1 will pop up offering you several fields and pop-up menus to specify important information about the new story. The top of the dialog simply presents a lot of identifying information about the story (including its internal unique identification number, the name of the author, and when the story was created); this is followed by a line the give the story’s title. The
middle section of the dialog contains three fields and two menus. Their use is described in the following paragraphs. The bottom section contains two buttons for exiting the dialog: click "OK" when you want the system to store the information you have specified; click "Cancel" to have the system ignore any information you may have entered.

Summary field: The large unlabeled field is provided for you to type in a one sentence summary of the story. This summary will appear in the annotations margin of the Design window when the story is linked to the current design display and is part of the Story pane's selected set. The box is sized so you will know how long a summary will fit in the Design window.

Picture field: This field allows you to enter the name of a file that should contain a PICT to be displayed along with the story text as an illustration. The naming and storage of PICTs is a little tricky and will be discussed below in the section on "Illustrations".

Movie field: This field is intended to work the same way as the Picture field, but for QuickTime movie files rather than PICT files. At present, it is not yet implemented.

Class menu: Clicking on the box with the black triangle next to the label "Class" will produce a pop-up menu, from which you
can select one of three story classes: **Point**, **Interaction**, or **Cluster**. At present, the system makes no use of this information, so you probably don’t have to worry about it.

**Outcome** menu: Clicking on the box with the black triangle next to the label “**Outcome**” will produce a pop-up menu, from which you can select one of three story classes: **Positive**, **Negative**, or **Mixed**. Unlike the class information, the system does actually make some use of this information (though not yet enough!). The outcome setting determines the color of the dot that will be used to indicate when the story has been linked to a design display: white indicates a positive outcome; black a negative outcome; gray a mixed outcome.

Select “**Story Interests...**” from the Story pane’s **Modify** menu: A new window will appear that looks almost identical to the Interests pane of the Lessons window. This window, however will come up completely blank for a new story (it will not even display a blank interest form). You must create each new set of interests for your story explicitly by using the **Choose** menu. Initially, your only option will be the item labeled “**New**”; once you have an index on display in the window, the “**Copy**” item will also be available. Choose “**New**” to get a blank interests form, then fill it in just as you would in the Interests pane (see the section “Specifying Interests” in chapter 3).

Establish links for the new story: This will be discussed at length in a “Linking” section below.

**Copying an existing story**

To add a new story to a CBDA that starts out as a copy of an existing story (the assumption here is that you are going to then modify it so it is different in some important way), you should go through the following steps:

**Display the story you want to copy in the Story pane:** Use the **Choose** menu or other search and browsing techniques to get the existing story you want to copy displayed in the Stories pane.

**Select “**Copy Story**” from the Story pane’s **Modify** menu:** A duplicate of the currently displayed story will be added to the selected set in the Story pane (and to the system’s library of stories). The story will be given a title indicating it is a copy of the original story. Editing is automatically allowed on this new copy story.

**Make any necessary modifications to the new copy story:** See the next section for details of how to modify an existing story.
Note that the new copy story will not be assigned all the same links as the original story of which it is a copy. Nor will it be assigned the same set of interests. Copying only duplicates the visible fields of the Stories pane and the invisible fields of the “Story Fields...” dialog.

Modifying an existing story

To modify an existing story in a CBDA you use essentially the same techniques as you would use to create a new story and initially fill it in. There are a couple of initial preparatory steps however:

**Display the story you want to copy in the Story pane:** Use the Choose menu or other search and browsing techniques to get the existing story you want to copy displayed in the Stories pane.

**Make sure the “Allow Editing” item is checked in the Story pane’s Modify menu:** This item must be checked for any modifications to be allowed. It is automatically checked for any stories that you create during a session with the CBDA. You must check it off explicitly for all other stories. Note that the program remembers for the duration of your session all the items for which you have allowed editing. Thus, for instance, if you return to modify an item you had created earlier in a session, you will still be allowed to edit it.

Note that the window displayed in response to the “Story Interests...” menu item can be used to modify the set of interests assigned to a story. Not only can individual interests be changed, but options on that window’s Choose menu allow for the creation of new interests (using “New” or “Copy”) and the deletion of existing interests (Using “Delete”).

Deleting an existing story

To delete an existing story from a CBDA go through the following steps:

**Display the story you want to delete in the Story pane:** Use the Choose menu or other search and browsing techniques to get the existing story you want to copy displayed in the Stories pane.

**Make sure the “Allow Editing” item is checked in the Story pane’s Modify menu:** This item must be checked for any modifications to be allowed. It is automatically checked for any stories that you create during a session with the CBDA. You must check it off explicitly for all other stories. Note that the program remembers for the duration of your session all the items for which you have allowed editing. Thus, for instance, if you return to modify an item you had created earlier in a session, you will still be allowed to edit it.
Select "Delete Story" from the Story pane's Modify menu: When you try to delete a story a dialog box pops up asking for confirmation. Click on "Yes" if you really want to delete the story (or simply type return). Click on "No" if you want to cancel the deletion.

Confirm the deletion in the dialog box: The currently displayed story will be removed from the system's library of stories (and from the selected set). Some other story from the current selected set will be displayed (if there are any others).

Modifying problems

The procedures for creating, copying, modifying, and deleting problems are almost identical to those for stories (except of course that all operations are performed in the Problems pane rather than the Stories pane). The major differences include the following: 1) problems are general statements not associated with any particular project, thus they have no project field; 2) the dialog box called up by the "Problem Fields..." menu item differs from the analogous dialog for stories in that it is a bit simpler (right now, the only field that really matters is the Pict field); and 3) the kinds of links that make sense for problems are far different than those for stories (modification of links is discussed in its own section below). See the discussion of "Modifying stories" directly above.

Modifying responses

The procedures for creating, copying, modifying, and deleting responses are almost identical to those for stories (except of course that all operations are performed in the Responses pane rather than the Stories pane). The major differences include the following: 1) responses are general statements not associated with any particular project, thus they have no project field; 2) the dialog box called up by the "Response Fields..." menu item differs from the analogous dialog for stories in that it is a bit simpler (right now, the only field that really matters is the Pict field); 3) the kinds of links that make sense for responses are far different than those for stories (modification of links is discussed in its own section below); and 4) responses are not indexed using elaborate descriptions, so there is no "Response Interests..." item in the Modify menu of the Responses pane. See the discussion of "Modifying stories" directly above.

Modifying sources and their annotations

The procedures for creating, copying, modifying, and deleting sources are somewhat similar to those for stories (though of course that all operations are performed in the Sources pane of the Sources window rather than the Stories pane of the Lessons window). Sources have a significantly different structure
than stories: they have many more visible fields (Title, Author, Year, Volume, Pages, Publisher, and City, all explicitly labeled in the pane), and they have no invisible fields. The many visible fields require use of standard Macintosh text field techniques for moving among and fill out text fields. The lack of invisible fields means that the Modify menu of the Sources pane has neither a "Source Fields..." nor a "Source Interests..." item.

Each source can also have associated with it a set of annotations. These are displayed to the right of the source in the Annotations pane of the Sources window. An annotation is basically just a chunk of text describing the source, that is itself attributed to some user of the CBDA. Annotations are created, copied and deleted just like all other presentations, that is, using items in their pane's Modify menu. Annotations are automatically tied to the source that is visible at the time they are created; the annotations associated with a source always form the selected set in the Annotations pane when the source is displayed in the Sources pane.

---

Modifying designs and their displays

This is one of the major gaps in the current implementation. The only way to make most modifications to designs is to have define privileges and to directly modify the relevant data structures using the facilities of the Define window. The necessary operations will be discussed in the next chapter.

---

Linking

As described in the chapter on "Browsing", the Browse menu in each pane is the primary method for moving from a pane's current item to related items. The system keeps track of related items by storing links between pairs of items. With modify privileges, the Browse menu can also modify link structures. When "Allow Editing" is checked for a displayed item, holding down the option key while selecting from the Browse menu tells the system you want to create or delete a particular kind of link from that item.

The system responds to an option-click on a Browse menu by displaying the floating dialog box shown in figure 4.2. While this dialog is visible, you are still able to use the basic windows in the CBDA and move around in the system, however, the dialog always floats in the front of all other windows to remind you that you are in the middle of making a change to the link structure. The first line of the dialog notes which pane displayed the item that will serve as the source of the link, and which pane's currently displayed item will serve as a new link's destination when the "Create Link" button is clicked. The second line is a reminder of what item was on display in the source pane when the link was requested. In figure 4.2, the user option-clicked on the "Problem" item in the Story pane's Browse menu while the pane was displaying a story titled "A Sample Story with a Sample Title".

48
Request the deletion by clicking "Delete Link" in the dialog box. The dialog will disappear, but a new one will appear asking you to confirm the deletion.

Confirm the deletion in the new dialog box: Click on "Yes" if you really want to delete the link (or simply type return). Click on "No" if you want to cancel the deletion.

Creating design display links

Links from design displays to stories are a bit different from other links in a CBDA. They appear as small round dots on graphics in the Design window rather than as menu choices. These links are still created using the same linking dialog described above, but the way linking is initiated must obviously differ slightly. Instead of option-clicking on a menu item, you must option-click directly on the graphic; the position of that initial option-click determines where the link’s dot will appear.

Deleting design display links

To delete a design display link, option-click on the graphic containing the link to be deleted, and when the linking dialog appears use its deletion features as you would for any link type.

Moving design display link dots

To relocate the dot for a design display link, option-click on the dot, and drag it around the graphic to its new position.

Illustrations

Illustrations can be associated with any problems, responses, and stories. Figure 2.3 showed a story with an accompanying illustration sharing the Story pane with the story’s text. The connection between a story and an illustration is established in the "Story Fields..." dialog by filling in the Pict field; the analogous dialogs for problems and responses play a similar role. This section discusses the details of how that part of these dialogs works, and what else you must do to make sure the illustration appears.

The PICTs that accompany texts are one form of media (currently the only form supported by Design-MUSE). Media presentations such as PICTs must be prepared using other applications available on the Macintosh. Almost any graphics program that runs on Macintoshes can produce PICT files. Somewhat more specialized programs are required to produce QuickTime movies, the next media form likely to be added to this system.
The Data folder of every CBDA’s directory structure contains a sub-folder called Media. The Media folder contains sub-folders for each of the major CBDA presentation types that can have associated media; that means the Media folder contains folders called Problem, Response, Story, and Display. These folders, in turn, contain sub-folders for the major media types that can be paired with CBDA presentations; right now, the only sub-folder is called Pict. It is in these Pict folders that you should store the PICT files that you want to associate with CBDA presentations. Thus, if you have a PICT file that you want to use to illustrate a story, you should place it in your CBDA’s folder “Data:Media:Story:Pict”

The name of the file forms the final crucial link in associating the CBDA presentation and the media file. What you fill into the Pict field of the fields dialog is the name of the PICT file. If you have entered the name of a valid PICT file, that PICT should be displayed along with the text next time you display the presentation. If the file exists, but is not a valid PICT file, the CBDA may generate an error and drop you into the underlying Lisp system. If the file does not exist, nothing will happen; however, next time you bring up the fields dialog, you will see “?” displayed to the right of the Pict field.

There is another way to associate media files with CBDA presentations. This method relies on the fact that every object in a CBDA is assigned a unique integer as an immutable identifier. You can find out what this number is for a presentation by looking in the fields dialog for a parenthetical note of the form “(UID = nnn)” (where nnn is some integer; see figure 4.1 for an example, where it says “(UID = 639)”). Since this number is guaranteed to be unique to the CBDA presentation and never to change, it can serve as a stable basis for naming associated files. The system will automatically look for files in its media folders with names of the form <presentation-type>.nnn. So, for instance, a file given the name “story.639” and stored in the folder “Data:Media:Story:Pict” will automatically be associated with the story who’s unique ID number is 639. When a file with such a name exists, the fields dialog will display a “*” to the right of the Pict field.

Saving modifications

All modifications that you make to a CBDA while running the program are made to the in-memory copy of the data. For the changes you make to affect future sessions, you must explicitly save your work so that the database is changed on disk. You can tell the system to save the current database in one of two ways: 1) you may explicitly choose the “Save” option from the File menu in the menu bar at the top of the screen; 2) you may wait until you are quitting from the system, and then click the “Yes” button in the dialog box that asks you if you want to save data. See the notes in the “Overview” chapter about “Saving data” in the section on “Specific warnings”
Chapter 5. Defining

Introduction

This chapter covers Design-MUSE features that are only available to users with define privileges. This includes establishing the indexing vocabulary used for composing search cues, and performing arbitrary modifications to all data objects. It is intended to include the ability to define and redefine object classes, but in the current implementation classes can only be inspected; changing their definitions currently requires a foray into raw Lisp hacking.

In some future version of this manual, this section ought to contain explanations that make clear what kinds of object classes are needed in a CBDA, and should dwell in particular on the creation of index frames and index vocabulary. Guidelines for these activities are still under development as part of the ongoing CBDA research program. Instead, this section will give a quick once-over the various definition commands. This probably duplicates the material in the "Quick Start" chapter section on "Defining".

Users with define privileges have access to one additional window. This Define window is composed of three panes. The Fillers pane allows for the definition of the menu items available to the user as options for filling out the Interests and Descriptions forms. The Classes pane allows for the display of all object class definitions, including those that appear as the forms in the Interests and Descriptions panes. The Objects pane allows for the creation and editing of any type of data object in the system. At present, this is the way that new artifacts and their graphics are entered into the system. Figure 5.1 shows the Define window displaying a collection of MIDAS's internal data structures.
How to...

Viewing object classes

The Classes pane currently allows viewing of object class definitions. It displays them in a tabular form, with a numbered row for each of the class’s fields. The tables have five labeled columns:

- **Slot Name**: This column lists the name of each field.

- **Category**: This column contains a symbol indicating that the type of filler allowed or the field falls into one of three major categories: `built-in`, `class`, or `name`. Built-in types include some of the basic Common Lisp data types, most notably symbols, numbers, and strings.
Class types are any CLOS classes defined within the system so as to be displayable (and some day editable) from within the Classes pane. Name types include any of the *spaces* that can be displayed and edited in the Fillers pane.

**Type:** This column gives the exact type restriction for the field, and must indicate a type from category specified in the second column.

**List?:** This column always contains either a "yes" or "no". "Yes" means the field may take multiple fillers. "No" means the slot can accept only a single filler.

**Weight:** This final column controls how instances of the class will be matched, and only applies to classes that are matchable (that is, classes that inherit from the class "index"). A weight should be a small integer; in matching objects it is multiplied by the basic score of the field's match to give a final match score for the field. Comparing the weights assigned to different fields gives some indication of the fields' relative importance during matching. When this column contains a blank, it means that the field is not used in matching.

Note that a minor bug in the code for the Classes pane requires you to check the "Allow Editing" item in the Modify menu before the table will allow you to scroll.

---

**Editing object classes**

While, for the most part, the Classes pane is limited to viewing existing class definitions, it does already support two simple types of modification to those definitions. In particular, the user can toggle the "List?" column values between "yes" and "no" and change the values in the "Weight" column. Both of these operations are only possible when the Modify menu's "Allow Editing" item is checked. Also, editing is limited to those classes defined within the particular CBDA and those fields defined locally to the currently displayed type.

When "Allow Editing" is checked a set of highlights track the mouse when it is clicked and dragged over the class display table. The cell under the mouse is inverted, as are the labels of the cell's row and column. This mouse tracking is limited to displays of locally defined classes and the rows for locally defined fields; the mouse will not respond over displays of classes defined in the basic Design-MUSE system, nor over rows representing fields inherited from superior classes. To change a value in a cell, click and release the mouse while over that cell. This currently works for "List?" and "Weight" cells. Though the mouse is tracked over other columns, nothing happens when you click on them.
Toggling a field’s “List?” cell

To change a field’s definition so that it accepts a list of fillers when it used to take only one filler, or to take only one filler when it used to take a list, simply click on the field’s “List?” cell. If the current filler is “yes” the a dialog will pop up asking to confirm the action and warning you of possible loss of data; turning a list field into a non-list field means that in any existing instances of the class, all but the first filler in the lists stored in the field will be thrown away!

Modifying a field’s “Weight” cell

To change the weight of a field in the match process, simply click on the field’s “Weight” cell. A dialog box will pop up asking you to enter a new integer. To make the change, enter an integer into the dialog’s blank field and then click the “OK” button or type return. To cancel the change, click on the “Cancel” button. Note that a bug in the current system means that the change in weight will not affect the matching routine until the system has been saved and restarted.

Creating, copying, and deleting classes

The Modify menu of the Class pane contains items that fit the standard pattern: “New Class”, “Copy Class”, and “Delete Class”. Unfortunately, in the current system, these items are not implemented. Even worse, the menu items are still often enabled. Worst of all, instead of getting a warning dialog box, the underlying Lisp system may generate an error. This situation was discussed in the notes on “Unimplemented commands” in the “Specific warnings” section of the “Overview” chapter. Don’t panic. Just ignore the error message and the window in which it appears.

Choosing object classes

The selected set for the Classes pane always includes all classes defined in a way that the system can recognize. While the Choose menu contains the standard options for moving through the selected set, the behavior of its “Pick One...” item is a bit non-standard. Instead of displaying the standard chooser dialog with a simple scrolling list, it displays a chooser dialog with a twist-down scrolling list reflecting the hierarchical structure of class definitions. The twist-down list works similarly to the Macintosh System 7 file displays. The left margin may contain a series of small triangles. A right-pointing triangle can be clicked on to reveal additional choices -- a set of items hierarchically beneath the item next to the triangle. When clicked, the triangle changes to point downwards and the nested items are displayed so that they can be selected. When clicked again, a downward-pointing triangle
turns back into a right-pointing triangle, and the nested items are removed from the list again.

**Viewing object instances**

The Objects pane allows you to both view and edit instances of the CLOS classes defined so as to be available in the Classes pane. The selected set for the Objects pane always consists of all instances of the class currently on display in the Classes pane. Instances are displayed in a format much like that used in the Interests and Descriptions panes -- a two column scrolling list, where the left column contains the field label and the right column contains a box displaying the filler for the field. Fields with multiple fillers appear repeatedly in the list, once for each filler. Any field that contain an instance of some CLOS class has a twist-down triangle in the left margin which allows you to expose the contents of the nested object; fields that are empty, or that contain a built-in data type or a name drawn from some space do not have these triangles (see the discussion of the “Category” column in the section above on “Viewing object classes”).

Note that a minor bug in the code for the Object pane requires you to check the “**Allow Editing**” item in the Modify menu before it will allow you to scroll the list of fields and fillers.

**Editing object instances**

To edit an object instance it must be on display in the Object pane, and the pane’s “**Allow Editing**” menu item must be checked. Then editing proceeds much as in the displays of the Interests and Description panes. There are two significant differences from what was described in the section on “Specifying interests”:

Not all filler fields have pop-up menus: Some of the boxes in the right column may not have the usual downward pointing black triangle that indicates a field has a pop-up menu. These fields take fillers that cannot reasonably be arranged in a menu (fillers such as numbers, symbols, or text strings). Clicking on these boxes brings up a dialog item appropriate to the kind of filler allowed for the slot. Generally, the dialog will be pre-loaded with the current contents of the field (if there are any).

**Fields of nested object can be edited when their twist-down arrows point downward to expose their contents:** As mentioned above, those fields that contain instances of some object class appear with twist-down arrows in the left margin, allowing you to expose the contents of those nested objects. When nested objects are visible, they are also editable using the same point and click conventions.
Note that these two additional points about editing objects are actually \textbf{not} unique to the Object pane. The same situations can arise in the Interest and Description pane, but these features were not discussed in that section because they do not come up in any current system.

\textbf{Creating, copying, and deleting object instances}

The \texttt{Modify} menu of the Object pane contains items that fit the standard pattern: \texttt{"New Object"}, \texttt{"Copy Object"}, and \texttt{"Delete Object"}. Unlike in the Class pane, these menu items actually do what they are supposed to do. You will have need of these facilities to take care of some chores that are not otherwise provided for in the more friendly interfaces in other parts of the system. For instance, there is no other way to create new users for the system than to select the User class in the Class pane, and then use \texttt{"New Object"} in the Object pane to create a new instance of the User class. That new user instance can then be edited as described above so that it has the right name and password.

\textbf{Viewing fillers (spaces)}

The Fillers pane supports display and editing of \textit{spaces}, which are data types peculiar to CBDAs. A space is a data type much like Pascal’s enumerated types; each space has some set of \textit{names} that are the valid objects of that type. (This is why, in the Class pane, a field with \texttt{Category “name”} requires some space as its \texttt{Type}.) Names are provided so that the definer of a CBDA’s representation system can, as is often necessary, invent new primitives. Names are grouped into spaces to reflect commonalties in their intended use -- the names in a space are often alternative values for some attribute, and are intended to be comparable to derive some sense of similarity and difference.

When compared to most systems allowing symbolic constants (such as Pascal’s enumerated types), the peculiarity of Design-MUSE’s names is that they are not really completely primitive -- that is, it is possible to assign them some substructure as a way to indicate graded amounts, or particular kinds of similarities and differences among names in the same space. A space, in fact, is defined by some set of \textit{discrims} and each name is characterized by some (usually unique) pattern of discrims. The idea of encoding recognizable concepts (here names) in terms of some underlying space of possibly uninterpretable micro-features (here discrims) is borrowed from connectionist systems.

When compared to connectionist systems using microfeature encodings, the peculiarity of Design-MUSE’s names is that we would prefer all our names and discrims to be semantically interpretable. While names are almost always meaningful (they usually appear in the interface on menus as choices for the user), the discrims may be more problematic. The idea is that
any distinction that might make a difference in judging similarity or
difference between two names drawn from the same space should be included
as one of the spaces discrims; then some names should include the discrim in
their definitions, while the others should not. Discrims are binary features,
and names are lists of the discrims that characterize the concept underlying
the name.

The major assumptions required to make spaces useful and workable are
1) that the items you want to talk about can be partitioned into distinct sets
(the spaces) where names in a space are comparable, and names drawn from
different spaces are incomparable, and 2) that the number of discrims needed
to define any particular space for the purposes of some target task or range of
tasks does not grow unreasonably large. What is “unreasonable” depends on
available hardware; the most common computations are logical operations
on bit-vectors, testing of particular bits, and counting of set bits in a vector. In
current technology where computers typically have 32 bit words, spaces with
30 discrims are trivial; soon, on 64 bit machines, 60 discrim spaces should be
equally efficient. The current implementation does not do a good job of
allowing visualization of spaces much larger than this anyway.

Matrix representation

The Fillers pane displays a space as a grid, with names down the left
margin, and discrims across the top. The definition of a name can be read off
by looking across its row for those discrim columns that contain black
squares. Patterns of similarity and difference between names can be read off
by comparing the patterns of black and white squares.

As a matter of general policy, it is convenient, and even makes sense that
each name have some discrim that is more or less uniquely its own. Effectively,
this discrim stands in for any unanalyzed residue of the name’s
meaning left over after describing it as some collection of otherwise available
discrims. In the degenerate case, a name has no more to its definition than
this single discrim standing for a completely unanalyzed meaning. A
consistent policy of inventing such discrims to accompany each name leads to
a grid display that contains a solid diagonal line of filled-in squares, perhaps
accompanied by some other pattern of filled squares. The Fillers pane, by
default, will create such patterns. Note that when all names in a space are left
as degenerate unanalyzed single-discrim patterns, the entire grid consists of
just this diagonal of filled squares.

Uses of filler classes

Spaces are used in several ways by Design-MUSE. First of all, patterns of
discrims are compared for similarity and difference; similarity is a numerical
rating determined by the number of bits shared between two patterns (and
also sometimes, but the number of bits they fail to share). Secondly, names
that stand for patterns of discrims often appear in menus offering choices to the user. Thirdly, the discrim patterns assigned to names are often used to induce a hierarchical structure in the menus; a name will appear in a nested menu beneath another name when the first name includes all the discrims in the second name plus some others. (Note, that a bug in the current system causes names that have identical discrim patterns to disappear from these menus entirely.)

The identity of space with a (possibly hierarchical) menu of names is probably a hack and of questionable value in the long run. As a potential plus, usability considerations for the menus have the nice efficiency effect of keeping spaces relatively small; an exceptionally well balanced hierarchical menu system, with about twelve items at each of three levels could organize about 2000 choices, but you rarely get that kind of balanced distribution, so the number of names you can comfortably organize in a space/menu is probably far fewer. Since the actual distribution of menu items is induced by the discrim patterns it seems unlikely you will never get such wonderful balance, certainly not without skewing the entire discrim encoding just to achieve that effect. This is one example of the general point that the two functions of spaces (representation and interface) may be at odds and perhaps ought not to be conflated as they are.

**Editing fillers (spaces)**

The Fillers pane does a pretty good job of providing a direct-manipulation interface for editing spaces. Of the six editing operations described in this section, only the first two live in the pane's Modify menu. The rest are invoked by mouse gestures on the spaces display grid. For any of these editing operations to work, the "Allow Editing" item must be checked in the Modify menu for the current space.

**Adding discrims**

To add a discrim to a space, select the "New Feature" item from the Modify menu. A dialog will appear, asking you to enter a new symbol to name the discrim. Once you type something, the "OK" button will become active, and then either clicking on it, or simply typing a return, will cause the system to try and interpret what you typed as the name of a new discrim. At any point in the feature dialog you can click the "Cancel" button to abort the creation of the new discrim. If a new discrim is created, it will appear as a new grid column at the far right.

The restrictions on discrim names are twofold: first you cannot use a symbol that already names a discrim in the current space; second, you must type a valid Lisp symbol (which essentially means, you cannot include any spaces, tabs or control characters in the name). If you do include spaces and
such in what you type, the system is likely to only read the characters that come before the blanks. If what you type (or what the system thinks you typed) is the same as an existing discrim name, the system will display a dialog notifying you of the conflict and will not create a new discrim. Also note that character case does not matter here; everything is converted to capital letters.

Adding names

To add a name to a space, select the "New Filler" item from the Modify menu. A dialog will appear, asking you to enter a new symbol to name the filler. Once you type something, the "Ok" button will become active, and then either clicking on it, or simply typing a return, will cause the system to try and interpret what you typed as a name. At any point in the filler dialog you can click the "Cancel" button to abort the creation of the new name. If a new name is created, it will appear as a new grid row at the bottom.

New names are always created with a discrim assigned to them. If the new name is the same as some existing discrim, then the name is created with that discrim filled in. If the new name is not the same as some existing discrim, a new discrim is also created and is filled in for the new name. The new discrim appears as a new grid column at the far right. The restrictions on filler names are the same as those on discrim names, as described just above.

Toggling a name’s discrims

A discrim is part of the definition of a name if the grid square at the intersection of the name’s row and the discrim’s column is black. Clicking on a black square to turns it white; clicking on a white square turns it black. The system interprets your action as a click when you release the mouse button while over a square. When you press down on the mouse button while over a grid square, the system highlights the square, as well as the square’s row and column labels (so it is easy to tell what name and discrim you would be modifying if you released the button at that point); the highlights will track the mouse so long as the button is held down and the mouse stays over the grid squares. To avoid toggling any square after you have already clicked down within the grid, just drag the mouse outside the grid and release the mouse button there.

Reordering discrims and names

To change the order in which discrims appear in the table, simply click down while over the discrim’s column label, and while holding the mouse button down, drag to a new position, releasing the mouse button when it is over the label of the column you want the original discrim to move past. If you move a discrim to the left, it will end up being placed to the left of the column where you release the mouse button; if you move a discrim to the
right, it will end up being placed to the right of the column where you release
the mouse button. The discrim you click down on will highlight, as will the
discrim columns you move over as you drag the mouse. To avoid moving
the original discrim after you have already clicked on its label, just drag the
mouse over the grid squares and release the mouse button there.

The analogous procedure works to change the order of names, but here
you manipulate the row headers and drag up and down rather than left and
right.

**Relabeling discrims and names**

To relabel a discrim, simply click down while over the discrim’s column
label, and release the mouse button while still over that label. The same
discrim labeling dialog will appear as would come up where you creating a
new discrim; the only difference is that, instead of coming up blank, it already
has the current discrim label filled in and selected. Click the “OK” button or
type return to change the discrim’s label after editing the contents of the field.
Click the “Cancel” button to avoid making any change to the discrim’s label.

The analogous procedure works to change the label of names, but here
you click on a row header.

**Deleting discrims and names**

To delete a discrim, simply click down while over the discrim’s column
label, and while holding the mouse button down, drag beyond the first of last
column label and release the mouse button there. The discrim labels will
highlight as if you were attempting to move the original discrim. You can
cancel this operation just as you would a request to move a discrim, by
dragging the mouse over the grid squares and releasing it there.

If you do release the mouse while in the column label area, but to the left
of right of all the current labels, the a dialog box will come up asking you to
confirm that you really want to delete this discrim. Click the “OK” button to
finish deleting the discrim; click the “Cancel” button to avoid deleting the
discrim.

The analogous procedure works to delete a name, but here you drag a row
header out of bounds. Note that since names are used as fillers in the CBDAs
data structures, deleting a name may remove data from arbitrary points in the
system. The system scans its data and the deletion confirmation dialog warns
you how many times the name you are about to delete has been used. If you
go ahead and delete the name anyway, the system takes care of wiping out all
those references to the now nonexistent name for you. Note this operation is
not undoable!
Choosing and creating, copying, or deleting fillers

The selected set for the Fillers pane always includes all available spaces. They are kept in the order in which they were created. The Fillers pane Modify menu provides the options "New Fillers", "Copy Fillers", and "Delete Fillers". These allow for the creation, copying, and deletion of entire spaces. A new space or a copy of an existing space requires a new title, which is requested by a dialog box. Deleting a space requires confirmation which is also requested by a dialog box.
Introduction

Sometimes there is no substitute for Lisp code. Sometimes you need to extend the system. Sometimes, you must work around an unimplemented feature or an outright bug. This chapter is intended primarily to help the savvy user work around the limitations of the current implementation. Until it is fully written, and even then, when the user is not comfortable with Lisp, expert assistance will be required to accomplish some basic operations.

Unfortunately, the Design-MUSE code lacks exemplary documentation. Nor is the code as well organized as it might be (progressive degeneration is the fate of all code, it just comes quicker to evolving academic prototypes). The system is at the point now where it would benefit from a major clean-up and partial overhaul. This would make further extensions easier. We may not, however, have that luxury, as other programmers are already beginning to study the existing code in order to use it as a base for their own work; pulling the rug out from under them would be bad manners.

How to...

Creating a new CBDA

To create a new CBDA application using Design-MUSE, there are several steps that must be taken before you even have an empty CBDA to run. First, you must create an appropriate directory structure for the new CBDA. Then you must set up several crucial files: the code file load.lisp, and the data files classes.lisp, space.lisp, and users.lisp. You also must set up empty versions of all the other data files required by your system. Finally, it is useful to start up Lisp, load everything for the new system, and then dump out a new Lisp image that will start up immediately as the new CBDA application. The following sections describe in detail what must be done.

Directory structure

The directory (Macintosh “folder”) structure for CBDAs built using Design-MUSE is a bit complicated, but the idea is to keep everything sorted
First of all, there is a major directory called **CBDA** that contains all the pieces of Design-MUSE proper. There are also separate directories for each CBDA application built using the shell (e.g., **Archie** and **Midas**). All of these top level directories have pretty much the same structure: they contain four sub-directories called **code**, **data**, **docs**, and **fasl**. Not surprisingly, these sub-directories hold, respectively, the systems' code, its data and resources, its documentation, and finally, the compiled versions of code files. The **code** directory should contain only `.lisp` files, the **fasl** directory only `.fasl` files (compiled lisp code files), and the **docs** directory only documents describing a system. This text you are reading now, for instance, belongs in the **cbda:docs** directory.

The **data** directory is quite complex. It is broken into four sub-directories: **media**, **objects**, **resources**, and **user**. The **media** directory organizes files that store data in standard Mac formats, such as PICT and QuickTime, and files created by other applications such as AutoCAD (so far, it only contains PICT files). The **objects** directory contains files storing external representations of the system's Lisp objects, and should probably eventually be replaced by some kind of real (object-oriented) database. The **resources** directory contains files that supply things like icons, cursors, background PICTs and other resource items. The **user** directory (not yet in use) is intended to store user-specific data, so users can, for instance maintain a notebook between system invocations (this probably ought to be pared back to some kind of preferences directory, so that users are free to maintain notebooks like normal documents, stored wherever they like). The **objects** and **media** directories require more discussion.
The objects directory is a primitive Lisp CLOS database. Files contain representations of CLOS objects, grouped by class. The basic format is Lisp forms that reconstitute objects in memory just as they were in some earlier session. The object directory contains three sub-directories: bkup, fasl, and lisp. The bkup directory exists to store one past generation of the data; since this data can be modified within the CBDA, users are likely to appreciate the system keeping automatic backups. The fasl directory stores compiled versions of the basic data files stored in the lisp directory. It is strongly recommended that you supplement the system's own maintenance of data backup files by frequently copying the lisp directory (or the entire data directory) to some safe location.

Finally, the media directory contains two more levels of directory structure: it immediately contains directories corresponding to major data classes (such as problem, response, and story), and those directories contain sub-directories corresponding to major media types or applications (such as PICT, QuickTime, AutoCAD). These directories then finally contain the actual media files. Note that somehow, the lisp objects (stored in the objects directory) have to keep track of which media files correspond they own.

The code file "load.lisp"

There is only one pure code file required to define a new CBDA, and that is the file load.lisp which should reside in the code directory. For now, the best way to create this file is to copy and edit the version used to define an already-existing system such as MIDAS. Figure 6.2 shows the load.lisp file for MIDAS. All lines beginning with semicolons are comments. There are really only six forms in the file. What follows is a brief discussion of what they do. This discussion will only make sense to someone who knows Common Lisp.

1. This first in-package form is needed because even though the file's mode line says the code in this file is to be evaluated in the MIDAS package, the second form in this file actually creates the MIDAS package. In a load.lisp file for some other CBDA you would have to change the mode line to mention the new <cbda-name> instead of MIDAS. You would also change all the comments that refer to MIDAS.

2. The defpackage for MIDAS mentions four other packages, only the first two of which are standard parts of MCL. The OOU package is a set of utilities created by Mike Engber when he was at Northwestern Universities Institute for the Learning Sciences; Design-MUSE depends heavily on the OOU package for much of its interface. The CBDA package is established by the Design-MUSE code itself. In a load.lisp file for some other CBDA you would have to change this form to call (defpackage <cbda-name> ...).
Figure 6.2. load.lisp File for the MIDAS CBDA.
This second \texttt{in-package} form makes sure that the remaining forms in the file get loaded into the newly created \texttt{MIDAS} package. In a \texttt{load.lisp} file for some other CBDA you would have to change this form to call \texttt{(in-package \texttt{<cbda-name>})}.

The \texttt{deffcbda} form does most of the work of defining the new CBDA. The contents of this form will be discussed in more detail below. In a \texttt{load.lisp} file for some other CBDA you would have to change this form to call \texttt{(deffcbda \texttt{<cbda-name>} ...)}. Make sure that all of the code files for the new CBDA are compiled and loaded. The function \texttt{compile-midas} is created by the \texttt{deffcbda} form. In a \texttt{load.lisp} file for some other CBDA you would have to change this form to call \texttt{compile-<cbda-name>}. In a \texttt{load.lisp} file for some other CBDA you would have to change this form to call \texttt{open-<cbda-name>}. Somewhere, before this file is loaded, you should establish a logical host with the same name as your new CBDA. If your CBDA's directory was stored in the \texttt{MCL 2.0} folder, you might put the following form in your \texttt{init.lisp} file.

\begin{verbatim}
(setf (logical-pathname-translations "<cbda-name>"
     (list (list '<cbda-name>::::*::* #p'ccl:<cbda-name>::::*::*))))
\end{verbatim}

The \texttt{deffcbda} form requires a fair amount of discussion. First of all, you should substitute your \texttt{<cbda-name>} for \texttt{midas} as the first argument. The first three keyword/argument pairs are largely cosmetic; they determine how the Notebook window cover will look. The trickiest one is \texttt{:cover-pict} which requires you to create a nice looking PICT file and store it in the right place. The easiest thing to do, once again, is to copy and modify the cover PICT for some existing CBDA. You should only modify the top third or so of the PICT, including the banner title and the illustration in the small box. The \texttt{:thanks} and \texttt{:credits} arguments should be self-explanatory.

The next block of keyword/argument pairs is more important and less intuitive. Each CBDA needs to be told which object classes are going to play each of six special roles in the system: project class, description-class, interest-class, problem-class, response-class, and story-class. (The system also needs to know which space is going to play a seventh important role in the interface -- that of view-space -- even though in the current implementation, the view-space ends up not being used!). Here we briefly describe what each of the six special object classes are for.
**Project-class:** This identifies the class whose instances will be taken as a complete artifact. The Design pane will root its partonomic hierarchy in instances of this class. It must be a subclass of the built-in class Project.

**Description-class:** This identifies the class whose instances will be used as artifact indexes and will appear in the Description pane. In the current version of the system, this must be the same as the project-class and the only searchable artifacts are entire projects.

**Interest-class:** This identifies the class whose instances will be used as lesson indexes and will appear in the Interests pane. It must be a subclass of the built-in class Interest.

**Problem-class:** This identifies the class whose instances will be displayed in the Problem pane. It must be a subclass of the built-in class Problem (or that class itself).

**Response-class:** This identifies the class whose instances will be displayed in the Response pane. It must be a subclass of the built-in class Response (or that class itself).

**Story-class:** This identifies the class whose instances will be displayed in the Story pane. It must be a subclass of the built-in class Story (or that class itself).

The final block of keyword/argument pairs identifies three sets of files that complete the system. A new CBDA can get along without requiring any extra code, so the code-files list is likely to remain empty. A new CBDA should not need any special resources, nonetheless, the rsc-file list should include the single file icons-dots anyway (and a copy of that file as used in the MIDAS system should be included in the "Data:Resources" folder for the new CBDA). Finally, the data-files list should be approximately the same as the one should for MIDAS, except that plane and sub-system should be eliminated from the list and the single file artifact used to replace them.

**The data file "classes.lisp"**

The file classes.lisp in the data:objects:lisp directory holds the definitions of all the CLOS object classes defined within a particular CBDA. Like all data files, it is kept up to date by the system itself. But also like all the other data files, it needs to start out with some basic stuff in it, that for now must be created by hand. The initial contents of this file should start out looking something like that for the MIDAS system as shown in Figure 6.3.

You will have to change the package to your own CBDAs new package. You will want to change the first defclass form to define whatever kind of
(in-package "MIDAS")

(defcbdaclass plane (project)
  
  ((togw :accessor plane-togw :type '(name togw) :initform nil :initarg :togw :weight 1)
   (customer :accessor plane-customer :type '(name customer) :initform nil :initarg :customer :weight 1)
  )

(defcbdaclass sub-system (artifact)
  ()
  )

(defcbdaclass midas-index (interest)
  ((system :accessor midas-index-system :type '(list (name system)) :initform nil :initarg :system :weight 1)
   (component :accessor midas-index-component :type '(list (name component)) :initform nil :initarg :component :weight 2)
   (issue :accessor midas-index-issue :type '(list (name issue)) :initform nil :initarg :issue :weight 4)
   (lifecycle :accessor midas-index-lifecycle :type '(list (name lifecycle)) :initform nil :initarg :lifecycle :weight 1)
   (stakeholder :accessor midas-index-stakeholder :type '(list (name stakeholder)) :initform nil :initarg :stakeholder :weight 1)
  )

Figure 6.3. classes.lisp File for the MIDAS CBDA.

artifact counts as a project in your system, and give it appropriate fields. For now, you can probably delete the second defclass form. You will probably initially just modify the name of the third defclass form to reflect the name of your new CBDA, but leave the five slots and their declared filler types until you develop a better idea of what kind of indexes you want to use for lessons in your system.

The data file "spaces.lisp"

The data file spaces.lisp can be created as a blank file that only contains a single form (in-package "<cbda-name>"). On the other hand, you may prefer to start it out with basic space definitions for all the spaces mentioned in the classes.lisp file (plus the space mentioned in load.lisp as the :view-space). The classes.lisp file in Figure 6.3 for instance mentions eight spaces, three in the class plane, and five in the class midas-index: togw, customer, mission, system, component, issue, lifecycle, and stakeholder. In addition, the sample spaces.lisp file from the MIDAS system shown in Figure 6.4 includes a definition for the space aircraft-view which is the system’s view-space.
(defnames aircraft-view)

(defnames togw)

(defnames customer)

(defnames mission)

(defnames system)

(defnames component)

(defnames issue)

(defnames stakeholder)

(defnames lifecycle)

Figure 6.4 spaces.lisp File for the MIDAS CBDA.
To adapt Figure 6.4 to your own use, you would have to change the package to your new <cbda-name> and edit the rest of the contents so you have defspace defname pairs for the spaces in your new system. I have left in MIDAS's fillers for the stakeholder and lifecycle spaces since those are likely to be of general utility (or at least to set a pattern you can follow). The only other point to note is that the second argument in the defspace forms controls how the names in the space will be ordered when they appear in menus. The two choices are :as-given and :alphabetical. In the current system, this option can only be set by going in and editing the space.lisp file.

The data file “user.lisp”

You should set up your system with at least one initial user and you should give that user define privileges. This requires hacking up an initial user.lisp file with the appropriate make-instance form. What you want is a file that looks something like in figure 6.5. As usual, you must substitute your <cbda-name> for MIDAS. And you should substitute your own initial user’s :title and :password.

```
(in-package "MIDAS")

(make-instance 'user
  :title "domeshek"
  :level #n(USER-LEVEL DEFINE)
  :password "eric"
)
```

Figure 6.5. user.lisp File for the MIDAS CBDA.

All the other data files

You must create initial blank files for all the other data files listed in load.lisp. All that should be in these files initially is a single (in-package "<cbda-name>") form.

Creating a CBDA application

To create a new CBDA and build it as an application you must first create the whole directory structure described above, then set up a load.lisp file, and establish the initial contents of all the necessary data files, all as described above. Then you must put the whole directory structure you have just created for your new CBDA into a directory called Work, along with the whole basic CBDA directory structure. That Work directory must be in the same directory as an MCL 2.8 image. You also need to establish the logical host for your new system and for the basic CBDA code, probably in your init.lisp file.
If your MCL 2.0 image has not already been specially prepared with all sorts of extensions preloaded, then the directory with Work and the MCL 2.0 image should also contain the standard libraries and examples directory that come with the MCL 2.0 release, plus the non-standard oodles-of-utils directory available by FTP. For now, one other directory called Contrib is also required, and it must contain a single file called passwords.lisp. This will eventually be merged into the basic CBDA code.

To build a new Lisp image with your new CBDA loaded in, perform these incantations:

1) Launch the Lisp image.
2) Type (load "cbda:code;load")
3) Type (load "<cbda-name>:code;load")
4) Type (in-package "<cbda-name>"
5) Type (save-application "<cbda-name>" :toplevel-function #'start-<cbda-name>)
6) Wait for Lisp to create the new image and exit.

You should end up with a new file in the same directory as your original Lisp image called <cbda-name>. This is a new Lisp image that can be double-clicked to launch your new CBDA. If you want, you can move this image and the Work directory to a new location, and you should be able to run the CBDA without the rest of the Lisp support files.

This whole exercise can take a considerable amount of time, particularly if your original Lisp image does not have all the support files preloaded, and if the CBDA code is not already compiled. Have patience. Good luck.

Modifying class definitions

As explained in the discussion of the Class pane, many of the features intended to make it easy to play with alternate indexing frameworks are not yet implemented. Until that deficit is fixed, it is often necessary to go in and edit the file classes.lisp by hand. Understanding exactly what to do is immensely easier if you already understand something about CLOS programming in Lisp. Nonetheless, we’ll take a shot at describing the relevant parts of the file, as pictured in Figure 6.3 above.

The bulk of the file is a series of defcbdaclass forms, which are slight variants on the standard CLOS defclass form. As formatted in Figure 6.3, the first line of each defclass form contains only the first two arguments:
the class name and a list of superclass names (those classes the new class will inherit from). You should probably not monkey with the superclasses. If you change the class name, you will also have to go to the data file that contains all the instances of the class and change the make-instance forms to use the new name of the class. If the class is mentioned for one of the special roles in the system's load.lisp file, you will also have to make the change there; note however that to have any effect, you then have to go on and create a whole new Lisp image for the CBDA (this kind of hard-wired dependency is a very bad idea, and will have to change in future releases!).

The next argument to each defcbdacllass form is a list of slot specifications, and it is generally spread out over several lines, with each slot spec itself a list taking a line or two in the file. Each slot spec starts with a name for the slot (which ought to be unique within the class and all the classes it inherits from) and is then followed by several keyword/argument pairs. Each slot spec should include the following pairs:

:accessor -- This keyword/argument pair is required. The argument should simply be a concatenation of the class name and the slot name, with a "-" placed between them (see Figure 6.3 and note that every slot is defined this way).

:type -- This keyword/argument pair is required. It is the most complex item required in specifying any slot. For the most part the type restrictions on slots will follow the uniform pattern apparent in Figure 6.3: "'(list (name <some-space>))". This means that the slot can take a list of fillers (as opposed to one single filler) and that each filler will be a name drawn from the particular space. You may drop the initial list (and its extra set of parentheses) if you like. you may also substitute some other valid Lisp type specification (e.g. a CLOS class defined within the system, or one of the following Lisp built-in classes: boolean string number integer symbol pathname.

:initform -- This keyword/argument pair is required. Generally the argument should be nil (as it is everywhere in Figure 6.3). In some cases, as when the slot :type indicates a number, it may be appropriate to initialize a slot to 0 instead of nil.

:initarg -- This keyword/argument pair is required. The argument should simply be the same as the slot's name, but with a colon prefixed (see Figure 6.3 and note that every slot is defined this way).

:weight -- The keyword/argument pair is not required. Only those slots that are to participate in matching need be assigned a weight. It is probably reasonable to start all such slots with a weight of 1. The facilities of the Class pane can then be used to modify this weight if necessary.
Changes made to the definition of a classes slots may also require going into the data files for instances of the class and modifying those appropriately (adding, deleting, or renaming slot initargs in the make-instance forms, or even changing the fillers to fit modified class restrictions).

The final arguments to the defclass forms are additional lists, each containing a single keyword/value pair. The ones evident in Figure 6.3 are the lines that say "(:indexexp t)". Again, you should probably not fiddle with these. You also should not need to add anything more complicated here so long as you avoid slots that have as their :type other CLOS classes. When you do introduce such slots, you should add a line that says "(:object <slot-name1> <slot-name2> ...)" for each slot that takes CLOS class fillers. Again, this can require major overhauls to the data files as well, so modification to existing slots of exiting classes is strongly discouraged when there are lots of existing instances. Keep your fingers crossed and wait for the Class pane interface that makes all this more transparent.