Collaborative Refinery: A Collaborative Information Workspace for the World Wide Web

Technical Report 97-03

David W. McDonald
Mark S. Ackerman
Department of Information and Computer Science
University of California, Irvine

The conceptual framework of a new system, Collaborative Refinery, is motivated by a scenario involving the creation of an FAQ. The scenario introduces the concepts of collecting, culling, organizing and distilling. Distilling is a specialized form of collaborative authoring with support for content selection and genre. The Web-based user interface supporting access to the four conceptual functions is presened in detail. As well, the system architecture and a current implementation are described in detail. Several research directions are discussed with respect to Collaborative Refinery. Collaborative Refinery is related to prior research in personal information management, collaborative authoring, and shared workspaces.

A common problem with large information sources, like the World Wide Web (WWW), Usenet news, or even a group memory system, is the difficulty of find useful information. The sheer volume of information prevents a novice from differentiating between the genuinely useful or interesting items and other less useful information. Summaries, abstracts and hot lists provide a novice instant entry points in a new topic area. The novice information seeker simply looks for these specific entry points like a Frequently Asked Questions (FAQ), a discussion list digest, or a Web hot list.

FAQs, digests and hot lists ease the burden on the information seeker. The burden is shifted to the FAQ or hot list creator. The burden of collecting information sources, organizing items from those sources into coherent topics, and creating a digest often falls on one interested individual or some local expert.

The Collaborative Refinery attempts to leverage the collaborative and incremental effort of many participants to create shareable information repositories. The creation of digests, abstracts, indexes, and FAQs represent a significantly different type of collaborative writing that should be studied.

The Collaborative Refinery relies on the World Wide Web and the user's local Web browser to present a shared workspace. The workspace allows collaborative browsing, searching, and modification by each user. Other systems attempt to support effective information sharing through automatic text processing techniques or artificially intelligent agents. Collaborative Refinery is an alternative approach that relies on existing human behaviors in collaboration.

The following discussion of Collaborative Refinery is motivated by a stylized collaboration scenario. The creation of an FAQ by a group of geographically distributed collaborators is the scenario. We consider FAQ creation to be a sophisticated set of independent, interrelated, iterative behaviors. These behaviors are too complex to capture in any static process description. The stylized scenario is a blatantly linearized process of FAQ creation to simplify the discussion. The Collaborative Refinery does not enforce nor require any static process model.

1. Collaborative Refinery Concepts

The Collaborative Refinery supports four generalized behaviors to effectively aid information pointing and seeking. The behaviors supported are collecting, culling, organizing, and distilling.

Consider the following scenario: You wish to create an FAQ of a news group along with some collaborators. The process of FAQ creation requires iterative modifications and refinement of approximate solutions to several related problems. For the sake of simplicity consider a stylized, linear process in which you and your collaborators solve the following:

- How do you gather and share the large quantity of information from which you will derive the FAQ?
- How do you categorize and organize all of the information in a way that is useful for creators and to later, subsequent, users of the FAQ?
- How do you decide what information belong "in" the FAQ and what will be left out?
- How do you abstract or digest the original sources to effectively present the FAQ contents?

The first problem in this scenario is the creation of an archive that the collaborators can share. This archive may consist of articles extracted from some existing archive as well as current, up-to-the-minute postings to the news group.

In Collaborative Refinery the creation of a shared archive is called *collecting*. Collecting gathers information to form an underlying repository. The process of collecting can vary based on the source and type of information as well as the intended use.

Collaborative Refinery supports several methods of collecting. Automated collecting is useful for certain information streams, like Usenet news or distribution lists. These streams can be selectively sampled through simple filtering techniques or they can be completely archived. Manual collecting strategies allow individual items to be submitted on a case-by-case basis. Manual collection happens through the system directly or through electronic mail. In this paper we call the result of collecting a collection or an archive.

Collecting an archive is an important part of the FAQ creation process. The next problem is organizing the content into manageable and coherent themes. You and your collabora-

tors would use the archive to identify recurring themes. Identifying key themes is one behavior that organizes the collaborators as well as the archive content and the resultant FAQ. Often some organizing effort is spent finding equitable workloads for each of the collaborators.

Identifying important themes and dividing the workload is not enough. Each item in the archive is evaluated in at least two ways. First, an item is evaluated to identify what theme or themes it contains. This evaluation groups relevant items together. An item that expresses two distinct themes might belong in two different groups. Second, the item is evaluated based on the significance of its content. The second evaluation is a judgement of the item relative to the other items with the same theme. The most significant items are included in the emerging FAQ. You and your collaborators might try to efficiently combine these two activities. You might try to identify themes and the significant examples of those themes in one fell swoop.

The Collaborative Refinery supports these two closely tied behaviors, *organizing* and *culling*. Culling identifies the most significant exemplars in an emerging collection, for example, a question, a clear set of instructions, portions of a discussion, or an answer. Another common example of culling takes the form of a best-of or alternatively worst-of list. Culling, therefore, is a selection mechanism in which people identify or highlight items contained in a broad-based collection. Items can be loosely culled to group them in a general classification scheme or tightly culled to represent well focused themes.

Identifying recurrent themes in a repository and linking items from the repository to these themes is *organizing*. There are many possible ways of organizing the content of any repository. For example, libraries commonly present organized projections of their content around subject, author, and title. Other common organizing schemes exist, such as organizing by date, frequency of use, or by priority. Organizing enhances the retrievability of sets of items in the repository by allowing information seekers to retrieve items using some established criteria. The combination of organizing and culling serves to enhance the retrievability of particularly interesting contents from the repository.

The last general problem in FAQ creation requires abstracting or digesting the culled and organized contents. You and your collaborators create synopses of the subtopics or threads from the news group. In the FAQ scenario, you would identify questions which are asked most often, pairing each question with a clearly worded answer. Often the question and answer is contained in a collection of posts. These posts must be carefully edited to form a fluid and coherent question and answer pair. The resulting question and answer pair will become part of the emerging FAQ.

In Collaborative Refinery editing or creating a summary, synopsis, or a set of pointers that are specific to a small portion of an organization (culled and saved subset), is known as *distilling*. The result of distilling is called a *distillate*. The terms distilling and distillate were chosen to convey the idea that the new document contains a more concentrated or concise form of the original information. These synopses are strongly related to the original information from which they are derived, often using excerpts, citing the original contributor and in some cases identifying the original source.

The FAQ creation scenario suggests support for four behaviors; collecting, culling, organizing, and distilling. The identification of distilling as a unique behavior represents an interesting conceptual contribution. Distilling can be considered from several points of view. The following section considers some distilling issues in depth.

2. Distilling: Types, Scope and Behavior

Distilling and the resulting distillates are interesting concepts that raise a number of questions.

- How does distilling compare to other forms of authoring?
- What is the range of distillate genres?
- How to identify the contents of a distillate?
- What are some important social components of distilling?

Each of these questions provides a different perspective from which to consider distilling and distillates. The following sections consider distillate authoring, distillate types, distilling scope, and social perspectives of distilling.

2.1 Distilling as Authoring

The previous sections describe distilling it in terms of editing, abstracting and digesting. As these descriptions imply distilling is a form of authoring. Distilling is a form of editing and the need to add, modify, and manipulate texts make it similar to many prior authoring systems. These prior systems often focus on supporting collaboration designed to generate original and unique texts.

Distilling recognizes the many types of editing that result in derivative works, such as digests, abstracts, or synopses. The requirements for supporting distilling are different than those for general collaborative writing. Commonly, derivative works identify their sources and recognize prior authorship more directly. Supporting the collaborative creation of derivative works focuses effort on the effective management of the original sources. Additionally, distilling may have more clearly defined types. The next section considers distillate types and how they may be supported by an authoring system.

2.2 Distillate Types

Distilling, like other writing forms, has distinct genres. We call these distillate genres *types* or *distillate types*. A general purpose collaborative authoring system may have difficulty supporting distinct genres. In contrast, distilling and distillate types may lend themselves to automated or assisted processing of the raw content.

Part of this work explores distinct distillate types and how those types can be supported. The types are loosely classified in two broad categories, *computer assisted* and *semi-automated* types. The computer assisted category encompasses types that require a large

amount of user editing and attention. Computer assisted types require a user to maintain the consistency and coherence of the distillate relative to changes in the sources from which the distillate was derived. The semi-automated types also require human intervention on their initial creation, but are maintained by the system for a period of time. The next sections consider specific examples of computer assisted and semi-automated distillate types.

2.2.1 Computer Assisted Types

Certain distillate types are tightly tied to the context in which they are distilled. Shifting context require subsequent changes in the distillate. Computing systems have difficulty detecting subtle shifts in context. One solution is to support distillate types that assist the user in the maintenance task, but that ultimately leave the user in control of the information content. Some distillate types that fall in this category include:

- Synopsis – Synopses are extended outlines of a discussion. A synopsis tries to effectively present, in a very short space, the essence of a point and counter-point discussion that may have no fixed resolution. Users who want more information than the distillate provides can always refer to the original messages from which the distillate was derived.
- Q&A – This type pairs specific questions with succinct answers. This type is most similar to many of the FAQs currently available.
- Tutorial – Tutorials have an instructional characteristic. In many cases tutorials will include a list of instructions or steps for a user to try. Tutorials often require merging a nearly correct prior tutorial with modifications or subtle changes to those steps.
- Narrative – Narratives tell a particular story relative to some set of distilled sources. The content of the narrative may not be based on any one of the original content items. This type may be more similar to an annotation that applies to the whole group of content items, rather than to individual items.

This is not a complete list of all computer assisted types. Ongoing research with Collaborative Refinery may identify types that are distinct from these and that have potential for computer assistance.

2.2.2 Semi-Automated Types

The semi-automated types differ from computer assisted types in the amount of processing that can be easily done to create a distillate. The semi-automated types allow for more of the work to be done by the distilling method. Users are still involved and they will still need to edit and maintain the content, but semi-automated types provide more specific assistance in the creation and maintenance of the distillate. The semi-automated types have been broken into two general categories called *pre-processed* and *temporally processed*. The categories are distinguished based on when the majority of the automated processing occurs.

Pre-processed types perform the majority of their processing before the user sees the rough distillate. These types have very specific forms which could be maintained by the system over some short period of time and over some reasonable number of additions. Examples of the pre-processed types include:

- Table Of Contents – A table of contents lists major topical ideas and concepts ordered in some meaningful way with references to specific groups of sources. An example of a table of contents distillate is one that simply points to other topically organized distillates.
- Glossary – A glossary presents key terms paired with their definitions in a succinct form with direct references to the source texts from which the definition was taken. This type is relative to a specific set of sources from which the distillate was derived.
- Index – An index is a list of key terms with references to the places where the term is used. This is relative to the specific sources from which this distillate was derived.

These examples are motivated by specific forms of automated text processing, but there may be other types that fit into this category. In these examples, the initial processing of the distillate could be extensive. The subsequent addition of source material requires an incremental update to the distillate that could be automatic. This update might not fit the exact intentions of the user, but is close enough that several additions do not destroy the coherence of the distillate. Eventually, after some number of additions the distillate needs some attention from a user for editing and maintenance.

On the other hand, temporally processed distillates only make sense in the context of system activity and the passage of time. That is, a temporally processed distillate must be processed and maintained in the context of the passage of time. Two examples of temporally processed distillate types include:

- Short Term Interest – There are many kinds of information that are only interesting for a short period of time. For example, most event announcements are only interesting for some small period of time before the event actually happens. Alternatively, an item that is listed 'for sale' is of interest for some short period of time, when either the item is sold or the seller ceases his attempt to sell it. A temporally based distillate could be set to watch for simple key words or dates, and if given the chance to execute at various intervals, maintained. Short term interest distillates have the characteristic that the source content, the announcement or the 'for sale' posting, could be removed from the archive once their time had passed.
- Frequently Accessed Questions – These types of distillates stem from the recognition that parts of a collection may be very active and other parts less active at any one time. It may be desirable to highlight either the least active or most active items. This type of distillate requires the system keep general purpose statistics over time. A user would set up the type of access behavior he wants to highlight and the system then maintains the distillate on the users behalf. One watching a distillate like this over time would see references to other distillates and other parts of a collection added and then removed as the activity changed. In contrast to short term interest type distillates, the source content behind frequently accessed questions would never be deleted.

2.3 Distilling Scope

Distillate creation includes two related problems. The first problem is that of determining the sources from which the distillate will be derived. This is a problem that the distillate author must resolve. In a hierarchy that is a mixture of potential sources of many different kinds, including other distillates, the distillate author needs an effective way to identify several sources from the many in the current hierarchy.

After the distillate has been created a similar problem exists from the reader's point of view. How does the reader of a distillate distinguish the sources from the many items in the distilled hierarchy? When a distillate is confusing, readers may require the sources to reconstruct necessary context surrounding that distillate. Additionally, readers need sources and authorship information to make quality judgements about the information contained in a distillate.

In Collaborative Refinery, culling and organizing are used to group a subset of items from a collection. Distillates are derived from a specific subset which is known at the time of distilling. Through this method, the distillate and the specific sources of the distillate are linked. This method requires the distillate author to select sources prior to distilling and allows the reader to know which sources contributed to a given distillate. This solution is limited; however a general solution for distillate scope remains a difficult and open problem.

In summary, this paper has suggested four conceptual features of Collaborative Refinery, collecting, culling, organizing, and distilling. Additionally, we have considered distilling and distillates in detail. Before describing how the architecture and detailed implementation supports these features, let us first consider how users see and access these features.

3. Using Collaborative Refinery

Collaborative Refinery was built to test our ideas about information refining and sharing. This implementation supports each of the four activities, collecting, organizing, culling and distilling with a shared workspace. Collaborative Refinery allows users to browse various representations of an information space, create new organizations, perform searches, and generate distillates. The user interface consists of a series of views or pages presented by a Web browser. Each view consists of a persistent button bar and a main view. The button bar displays functions which are available to the user at any time, while the main view presents forms, views, and controls that are specific to the current user activity.

HTML and the Web are far from an ideal interface for Collaborative Refinery. This application, like others which rely on the Web, can seem cumbersome and awkward. This awkwardness is a function of trying to implement the user interface as close to the basic Web standards as possible. We have storyboarded alternative interface options that may be utilized in subsequent versions of Collaborative Refinery, but that were not possible with the Web and HTML at the time of implementation.

The following sections briefly describe each of the main views and the functionality that is accessed through each view. Each view addresses some aspect of the FAQ creation scenario described above. Briefly, in the FAQ creation scenario you and a small group of friends are trying to create a FAQ based on a news group. The scenario suggests four basic activities necessary to create the FAQ.

1. Collecting - generating an archive of the news group that can be used as the raw material for the FAQ.
2. Organizing - determining the key themes present in the news group and grouping the items in the archive according to the themes.
3. Culling - identifying the most significant items from a given theme as most representative.
4. Distilling - creating a concise document based on the culled items for each topic.

The sections below describe how Collaborative Refinery supports these activities. For simplicity, the descriptions assume that a single user is attempting to access the specific functionality. The following sections contain figures which are screens from the working prototype. The content present in these figures is from a departmental bulletin board.

3.1 Browse View

Consider the basic problem of our user attempting to read some portion of the developing FAQ or some article from the news collection. The browse view supports the most basic functions of browsing collections, reading collection contents, and reading distillates.

When first entering Collaborative Refinery, the user starts in the browse view. Browse presents a hierarchical outline view based on the topics created by the various collaborators and items that have been added to the topics. Browse presents organized subsets of the collection. Browse facilitates collection management, distillate creation, and maintenance.

Figure 3-1 provides a sample browse view of a local bulletin board archive. In the figure, browse presents the user an outline of a hierarchical set of topics, contents and distillates. Topics that have either sub-topics or content below them are indicated with an expansion triangle preceding the node label. Clicking the expansion triangle expands the display to show all the existing items below the topic. If a topic has been previously expanded, clicking the expansion triangle will contract the display to hide all items below that topic.

Distillates are represented by a small document icon to the left of a topic item. Clicking on the distillate icon fetches the distillate and presents the text of the distillate in the browser.

Text labels represent either topics or content items in the current collection. Content items are indicated by underlining the text; topic labels are not underlined. Clicking on a content item text label fetches the item from the collection and displays it in the browser.

The browse view provides access to individual distillates and the contents of the collection. However, simple browsing is often not sufficient for a user to find any single item. Hierarchies with many topics and deep subclassifications can bewilder users and result in items that are lost in the classification scheme. The desire to provide information to help a user stay oriented in the organizational scheme results in the collection overview.

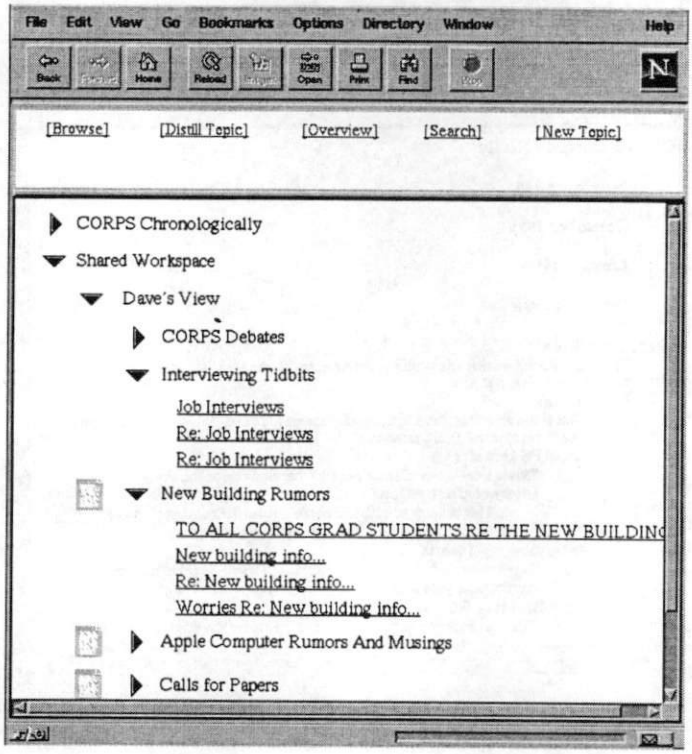

Figure 3-1 Browse view

3.2 Collection Overview

Hierarchically organized views can result in several problems when used collaboratively. One specific problem is knowing where in the hierarchy an item is located. Collaborators may attribute different meanings or different intent for the same portion of the hierarchy. The basic solution is to clearly communicate the intent of a given hierarchy to each collaborator. Formal organization schemes clearly communicate intent at the expense of flexibility.

Collaborative Refinery attempts to support flexible hierarchical views by providing a mechanism for communicating the intent of a given part of the hierarchy. The collection overview assists users by displaying the overall hierarchical organization along with the stated intent of the creator. With this view a user can quickly see the potential location of

an item. Additionally, this view provides clues for a user who desires to add to the growing hierarchical scheme. An example of the collection overview is provided in Figure 3-2.

The collection overview is accessed through the overview button in the button bar. This view displays a hierarchically organized list of the topics and descriptions of those topics. The topics, their hierarchical location, and descriptions are supplied by the users when creating a new topic, as described below.

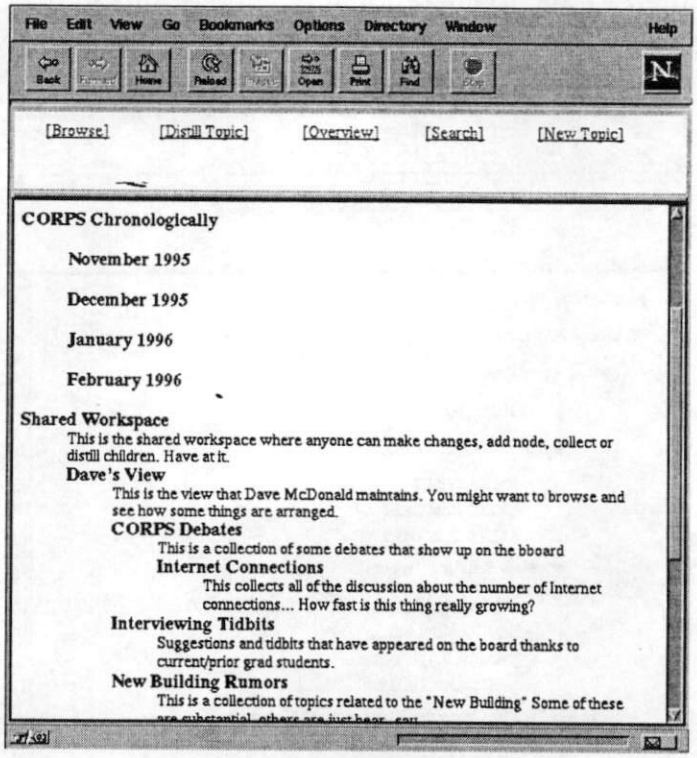

Figure 3-2 Collection overview

3.3 New Hierarchy Topics

In the FAQ creation scenario the collaborators identify recurrent themes or *topics* in the archive. These topics are used to organize the individual messages as well as the resultant FAQ content. Collaborative Refinery provides a means for users to create new hierarchically organized topics.

The new topic button allows users to create new organizing topics. The user is presented a form that has space for the new topic label and space for a description of the topic area. Users pick the location of the new topic in the current hierarchy through a choice hierarchy. Figure 3-3 is a sample of the new topic form.

The choice hierarchy presents the user a hierarchically organized list of current topics with a radio style button immediately to the left of each item. An example of a choice hierarchy is provided in Figure 3-4. The choice hierarchy represents an awkward, but

workable, solution to the problem of specifying item placement using HTML forms. The current choice hierarchy is a complete hierarchical layout of all of the topics in the current collection.

When the form is submitted the new topic is created. The new topic and the description supplied in the form will show up in the collection overview described above. Additionally, the new topic will show up in the browse view as an empty topic, with no content below it.

After identifying recurrent themes in the content, the collaborators in the FAQ scenario decide which of the content items are significant and should therefore be included in the FAQ. In Collaborative Refinery, as in the scenario, users must specify which items from the collection belong in any given topic area. Adding content items to previously created topic is called culling and is discussed below.

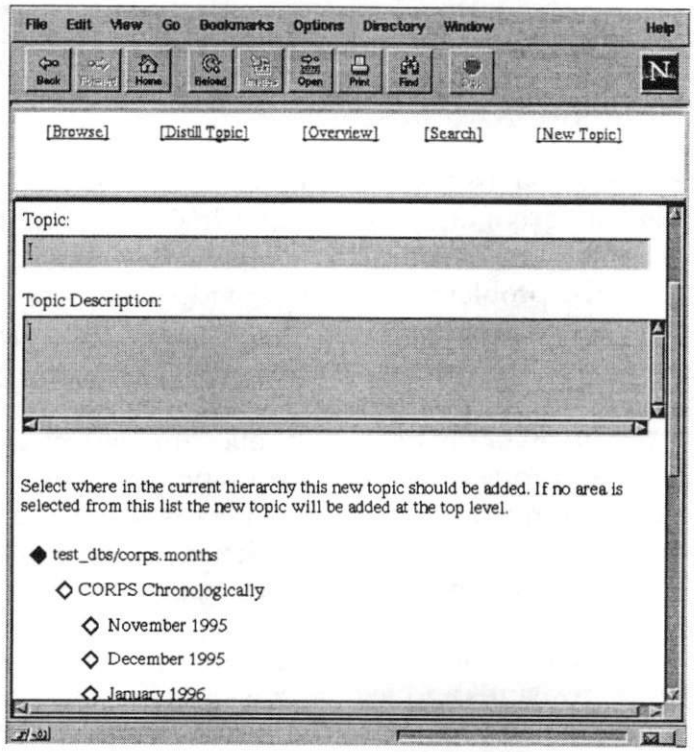

Figure 3-3 New topic view

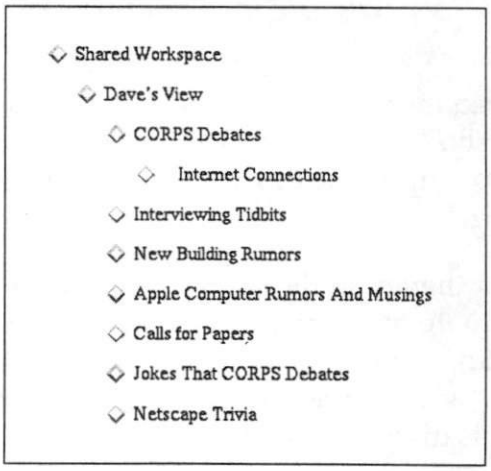

Figure 3-4 Choice hierarchy extracted from a dialog

3.4 Indexed Term Searching

Consider this simple problem: a user wants to find a specific item which is part of the collection, but which is not part of any browse view. In Collaborative Refinery the browse view displays a hierarchy of topics that includes distillates as well as culled and organized content items. The problem here is to provide users access to collection items which have not yet been culled and organized. A solution to this problem is to provide an alternative to the browse view, term searching.

Searching provides access to all of the items in a collection. Searching allows the user to enter between one and three terms with and/or conjunctions. The system then performs a term search through a system maintained term index. A sample of the indexed term search form is provided in Figure 3-5. The term index includes all of the content items in the collection, whether or not they are displayed in the browse view.

Items matching the supplied criteria are returned in an unordered list. Ordering the results through relevance measures and feedback is possible. Successful searches provide the user a list of content items as underlined text and topic nodes as plain text. Content items can be viewed by clicking the underlined text, like viewing content when browsing. Figure 3-6 shows an example of a successful term search with content items underlined.

A term search that provides access to all of the items of a collection leads to a simple solution to another problem. In the FAQ scenario, the collaborators must decide which items are significant enough to add to the current topics. The term search feature allows a Collaborative Refinery user to identify items with similar content and subsequently cull those items.

3.5 Culling and Categorizing

Culling is the identification of significant exemplars of a topic in the collection. Culling separates content items which should be included in the emergent organization from those that will be excluded. Users cull the archive contents through the search capabilities of the system. In Figure 3-6 the items that were returned by the search include checkboxes. The user marks an item as culled by clicking on the checkbox and then clicking the group button near the bottom of the page. This creates a culled subset of the current archive.

The culled subset is added to the topic hierarchy as children of some previously created topic node. The user is given a form that includes a choice hierarchy, like Figure 3-4, to indicate where the culled subset should be placed. Figure 3-7 provides an example of adding children using a choice hierarchy very similar to that used to add new topics.

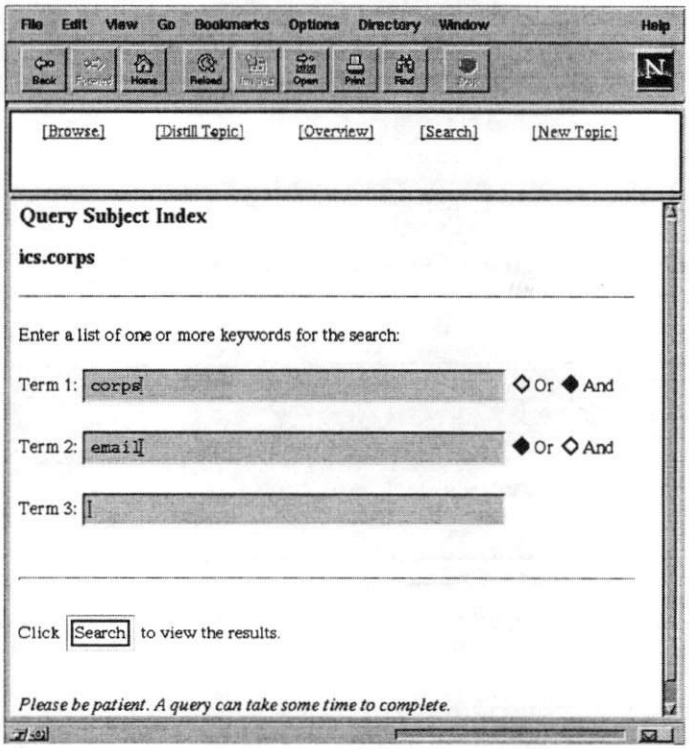

Figure 3-5 Term searching

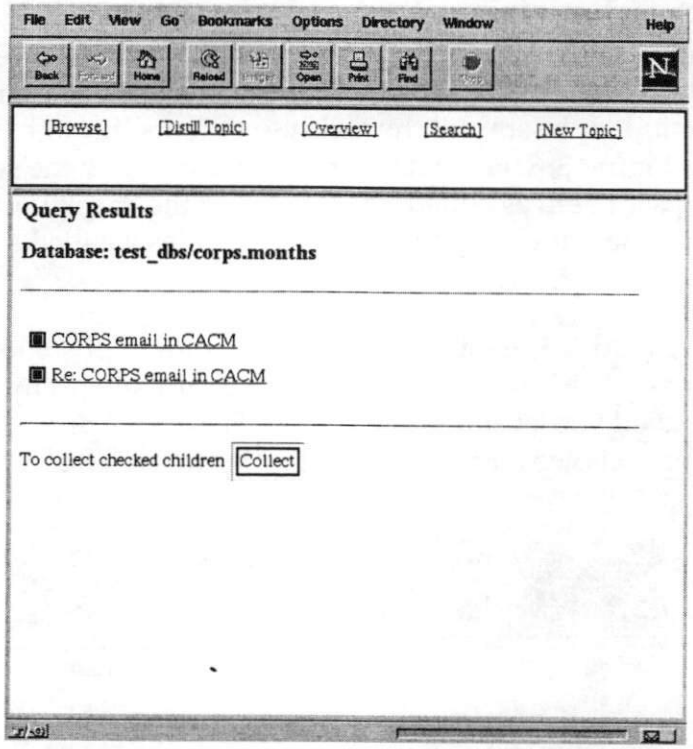

Figure 3-6 Search results and grouping

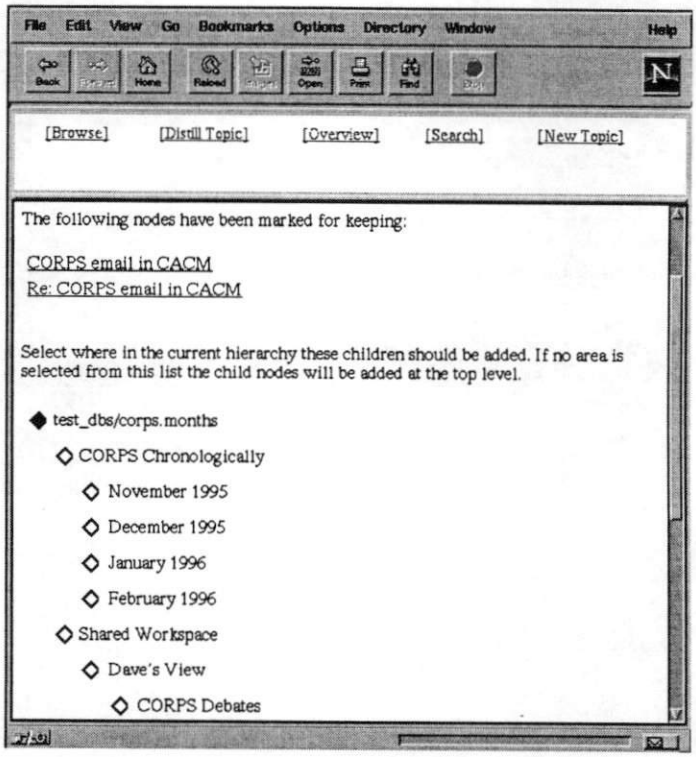

Figure 3-7 Placing a group

3.6 Distilling

The last major feature shown by the FAQ scenario is the creation or modification of a distillate. Collaborative Refinery currently supports a straight-forward version of distilling. The user creates a distillate by specifying both the content and resulting type of the new distillate. The content of a distillate is based on the content of a user selected topic node.

A user creates a distillate by clicking the distill button in the button bar. The user is presented a choice hierarchy and selects a node. The contents of the new distillate will be taken from the content immediately below this node and the resulting distillate will be attached to this node. Figure 3-8 is an example of the distill topic page.

With a selected topic node, there are several ways in which the content may be handled. Consider the general problem of maintaining an existing distillate. An author may update a distillate in different ways. The author may want to simply edit the distillate. Alternatively, the distillate author may want to add content based on recently archived and culled items. Another possibility is that the author may want to ignore the current distillate and start with all of the culled items in the selected topic node.

Accordingly, Collaborative Refinery supports four ways of selecting the culled content when distilling a topic node.

1. Use only new items - This option allows the distillate author to select items which are new to this topic node. In this context "new" means all of the culled content in the selected topic that was added after the last distilling.
2. Use both new and old items - This option specifies that all of the culled items under the currently selected topic be used for this distilling.
3. Use only the old items - This option allows the distillate author to include only the items that were available at the time of the last distilling. In this case the culled content items that were added since the last distilling are ignored.
4. Use none of the items - This provides a mechanism for simply ignoring all of the culled content. This is useful when the distillate author wants to edit the current distillate without adding any of the culled items.

These four options provide a simple mechanism for creating and maintaining distillates based on a specific topic node. In Collaborative Refinery, the mechanism for creating and updating a distillate are the same. First, the user must specify one of four options for the content under the selected topic node. The user must also specify wether the old distillate should be used as a basis for the new distillate. Lastly, the user selects a distillate type for the new rough distillate.

Distillate types are specified by picking a catalyst that will generate a specific rough distillate type. A catalyst is a Tcl script which produces a rough distillate that fits the combination of criteria supplied by the user. Catalysts are a simple mechanism that can extend the range of distillate types currently supported by Collaborative Refinery. Catalysts are covered in more detail below.

After a catalyst has run, Collaborative Refinery generates a form based on the rough distillate that the user will edit. An example rough distillate is provided in Figure 3-9. When the user is finished editing the distillate he submits the result and the system adds the new distillate to the proper node.

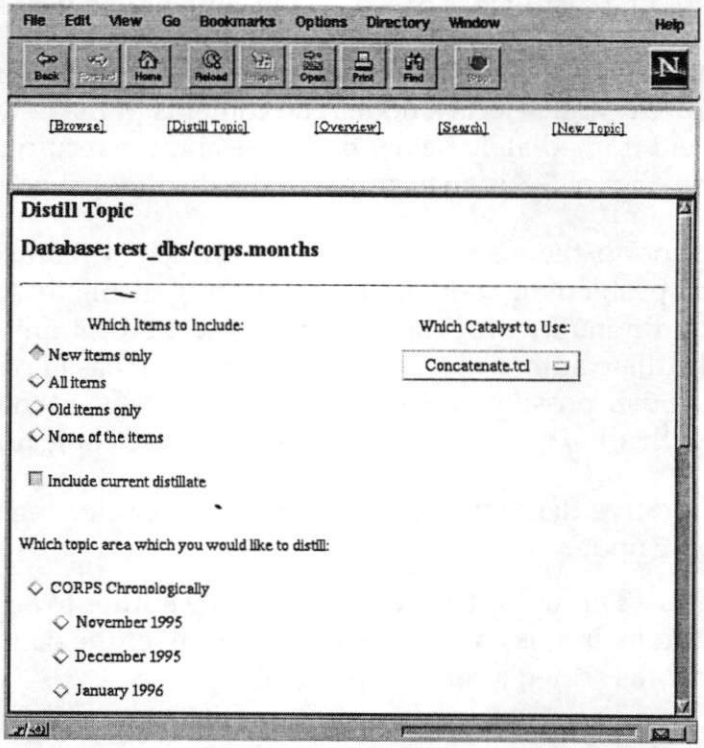

Figure 3-8 Distill a topic

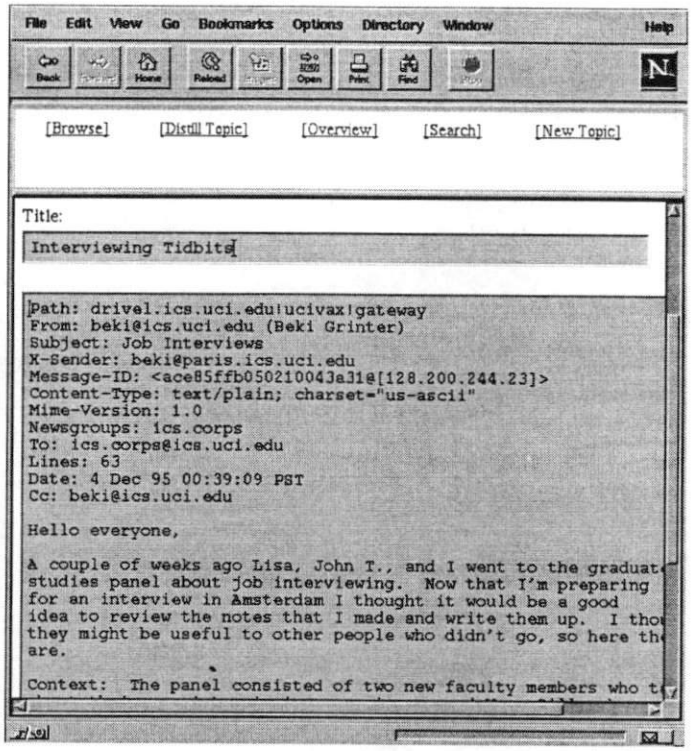

Figure 3-9 Rough distillate

3.7 Summary

A Collaborative Refinery user interacts with the system using five views — Browse, Overview, New Topic, Search, and Distill Topic. The Browse view allows the user to navigate the topic hierarchy, read distillates, and read culled content. The Overview allows the user to see the complete hierarchy as well as the topic information for each topic node. The New Topic view allows a user to create a topic node and place it in the hierarchy. The Search view allows a user to find content items that have not been placed into a hierarchy. Additionally, the Search view supports the selection of items and their placement into the hierarchy. The Distill Topic view allows the user to create or update a distillate with respect to a topic node. Through these five views Collaborative Refinery supports the refining process; collecting, organizing, culling and distilling.

The next section describes the system architecture and the implementation details that support each of the user views.

4. System Implementation

Collaborative Refinery is composed of three process components and four data components. The process components include:

- CGI Entry & Action Interpretation — This translates user actions into system activity. This component receives mouse clicks and form submissions from the local HTTP server and calls the database and presentation backend to generate a response.
- Database Operations — This process reads and writes nodes to a database, handles database queries, and manipulates the archive.
- Presentation Backend — This generates output for a browser by incorporating presentation specific markup into database query results.

The data components consist of:

- Archive — These are the items that will comprise a collection, stored as individual items in the file system. These could include email, news, topic descriptions, distillates or any other type of content item representable by the database and presentation backend.
- Database — A data store of abstracted information about each of the content items in the archive and the relations among the archive items.
- Intermediary Representation — A temporary representation of query results that the Presentation Backend can translate and markup in a single pass.
- Integrated Markup & Data — A temporary file which can be sent directly to the target viewer.

Figure 4-1 provides an overview of the system architecture and how the components interact. In the figure process components are denoted by circles and data components by rectangles. Arrows from one process to another represent a calling relation. Arrows between data components and processes represent a read or write relation. The grayed portions of the diagram represent items external to the system upon which the system relies.

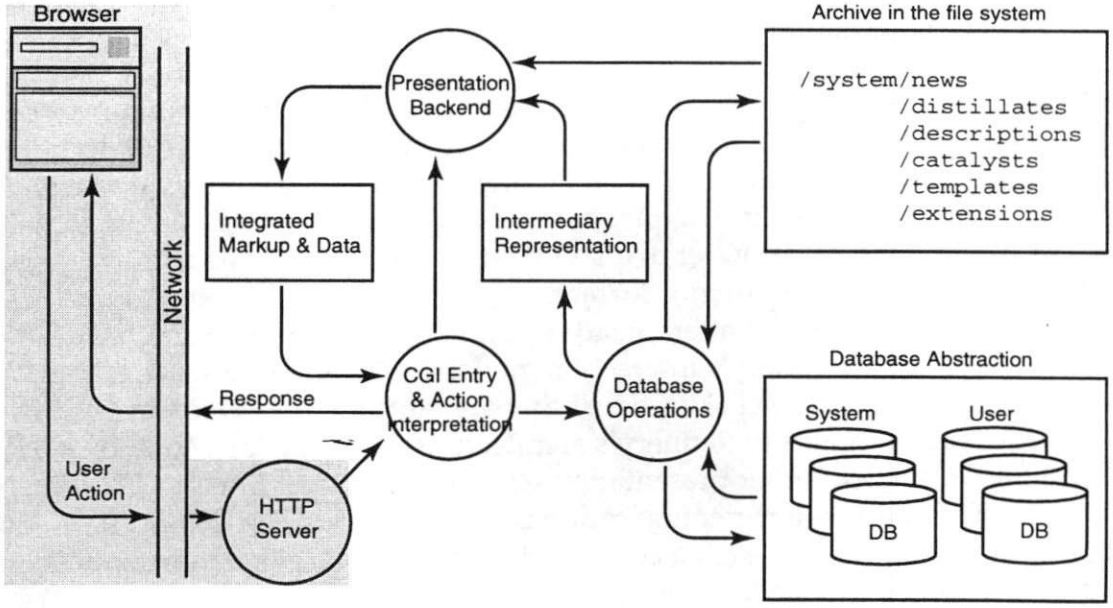

Figure 4-1 System architectural diagram

The archive, database and temporary components are stored in the file system of the server. A Common Gateway Interface (CGI) entry routine handles all of the user actions for the system. Depending upon the user action a database query is passed to the database process which generates an intermediary result. The entry routine then calls the presentation backend process to generate an appropriate response to the user. The user interface, as currently generated by the presentation backend, uses HTML 3.0 and browser extensions supported by Netscape Navigator 2.0 and several other browsers.

The sections that follow cover the architecture in detail. The first section discusses how the architecture supports a separation of concerns. The following sections cover the individual architectural components in turn.

4.1 A Separation of Concerns

The underlying system was designed to solve two issues while maintaining a shared workspace. First, the architecture separates the logical and physical structuring of the collections from their presentation. This separation allows simpler maintenance of complex relationships with the option of presenting subsets or slices of those structures as separate, presentable views.

By separating the presentation from the storage, Collaborative Refinery gains speed in manipulation of the data as well as flexibility in generating views of the data. This separation is maintained by combining an archive with a database abstraction. The database stores complex relationships as well as abstract presentation data to facilitate queries resulting in a low cost view. The archive allows queries requiring detail not maintained in the logical database abstraction. Users can satisfy detailed queries at higher cost by

following references to the archive items. Likewise, users can manipulate structural relations among collection items through database queries and modifications of database references rather than relying on manipulation of the archive items themselves.

Second, the architecture separates the presentation from the interaction model. The current version of the system provides a Web-based presentation with a Web-based interaction model. However, the system is not tied to the Web or its interaction model. (Presentation is how the results of a query are viewed by the user. The interaction model combines the way a remote browser displays the marked-up data and communicates the user interaction back to the system.) Currently, the implementation relies on HTML markup for presentation and it attempts to structure the presentation as specifically as possible. Collaborative Refinery's interaction model assumes a browser that supports form transactions and 'single click' actions[1] that are returned to Collaborative Refinery as HTTP requests. Community Refinery's architecture allows the presentation backend to be modified to support other presentation schemes such as text only or a system specific markup. The action interpretation routines that are part of the CGI Entry module could be easily replaced to present a command driven interface or a more sophisticated direct manipulation interface.

4.2 The Archive

The archive consists of semi-structured text messages such as news, email, or HTML pages. Items in the archive are stored as individual files in the file system. Additions to the archive happen either automatically, as in the case of the current Usenet news archiver, through direct user action (they run the programs that add items), or through interaction with the system interface. Currently, Collaborative Refinery includes a sample Usenet news archiver that automatically archives news groups and creates appropriate database changes.

Simply adding a file is not enough for the system to recognize a new content item. A representation of the new item must be added to a system database. The database, described below, must contain additional information for each item that will be represented by the system.

There are no logical restrictions on the types of items that can be placed in the archive. However, some types might require additional meta data to be effectively manipulated (e.g. audio, video). Additionally, some new types will require more sophisticated cata-

1. Current browsers have a very limited interaction model. Every user intention is encapsulated into some all encompassing 'click' of a mouse. This results from the original HTTP protocol and the way servers are extended through CGI. HTTP was intended to be a very simple protocol implementing a basic "GET" and "PUT" activity on some server. The implementation of this type of protocol in a user client only requires that the user indicate a single intention. Users only need to indicate that they want to get some remote item. The CGI extension mechanism is a hack on the "GET" HTTP request that allows some specified code to be executed. The result is that many user activities are forced into the single click paradigm.

lysts in order to create effective distillates. Adding new types requires modifying both the database and presentation backend code to recognize and present the new type.

The archive simply and effectively stores all of the content for Collaborative Refinery. However, as the previous section discussed, a means of separating the content data from the content structural relations must be provided. This separation is facilitated by a database which maintains the structural relationships. The next section covers the database content and the operations that are allowed on that content.

4.3 The Database

The database or several databases provide the logical connection between the objects in the archive and their presentation by the system. The database abstracts every archive object into a node. Every Collaborative Refinery node has type and relationship information that allow different queries to generate various projections. These projections are displayed to the user after the presentation backend reads and inserts markup into the query results.

4.3.1 The Database Structure

A database is central to presenting and retrieving items in the archive. As described above, each item added to the archive must also have a representation node added to some system or user database. A database node stores a number of attributes about the item that it represents. Some of the attributes stored for each node include:

- Creator — The creator of the node, which may be different from the creator of the item which the node abstracts.
- Creation and modification dates — The date which the node was created and the most recent date at which the node was modified.
- Text label — A label that can be used to represent the item. This might be the subject line of a mail message or the title of a report.
- Type — The system specific node type. Some node types are system specific representing logical relations among other nodes, aliases to other nodes and other maintenance types. Other node types represent the types of items in the archive such as news, mail, url, or distillates.
- Action — This is a key to describe how the system should retrieve the item when the user requests it.
- Physical location — This is the physical location of the item in the host file system.
- Tag node references — These are references to nodes in the current database which are attached in some permanent way to this node. Items like topic descriptions and distillates are tag references.
- Number of children — The number of child node references that this node has collected.

- List of child node references — This is a list of nodes that have been collected under the current node. These references point to specific nodes in specific databases. These are different than tag references above, in that these allow node references from databases other than the current one.

These attributes provide enough information to construct, or reconstruct, any of the views supported by the system. Collaborative Refinery generates a view by tracing appropriate node references and making node relationships explicit for the presentation backend. The presentation backend can then generate a view given the display and interaction constraints. The presentation backend is discussed below in more detail.

Reference attributes provide the means of representing and modifying the organizational scheme. Tag and child references point to other nodes in the same or in other databases. References, like aliases, are small compared to the overall node data. This reference structure provides a highly flexible means of linking and structuring the various nodes in an organizational scheme.

The structuring and representation of the database are important, but the data must also be manipulated and modified for any useful application. The next section covers the access and modification operations that are supported by this database.

4.3.2 The Database Operations

The database is accessed through CGI helper applications. Currently, these applications provide two classes of operations on the database, query and modify. There are four types of queries that these applications can perform:

- query list - Given the current user state, the action just performed by the user, and the item on which that action was performed, this query returns a list of all types of nodes and their display relationships.
- query display - Given a user action and an action item, this query retrieves the item from the archive and displays it for the user.
- query topic - Given a user action, this returns a list of just topic nodes and their display relationships. This is very similar to the query list, however that query does not differentiate between topic nodes and other nodes.
- query term_index - Given a list of terms and a set of conjunctive or disjunctive relationships among them, this query returns a list of nodes that contain terms that satisfy the term list and its relationships. The results of this query are scheme independent, because a node may exist in zero or many organizations at the same time.

Queries are performed in the context of a single database, but the relationships built by the organization scheme may require access to other databases and archives that the system maintains.

The CGI helper applications also support two general modification operations. Modification is used here in the most general sense of adding or modifying a database or nodes in a database. The two basic modifications are:

- add topic - Given a new topic name, description, and location in the current organizational scheme, this operation adds a new node to the database. If the new node is at the root level of an organization scheme then the node is simply added to the database. However, if the node is added somewhere within a hierarchy the node is added and the appropriate parent node is updated.
- add child - Given a list of new child node references, and a parent node location, this operation modifies the parent node adding the specified list of children.

Like the query operations, these modifications are performed in the context of the current database and organization. Some modifications may result in access to other databases in the system and potential modification to those other databases.

The completion of a database operation results in an intermediary representation that is used to generate a subsequent user presentation. The next section covers the intermediary representation in more detail.

4.4 The Intermediary Representation

Each user action generates a database query that returns an intermediary representation of the query results. The intermediary representation supports a quick single pass markup for presentation.

The simplest example of an intermediary representation is one that retrieves an archive item for display. In this case, a prior view would have presented an icon or a title as a hypertext reference. A click on this representation causes a database query. The query generates an intermediary representation that is a reference to the item in the file system. The presentation backend can use the reference to read the archive item and generate a new view.

A more complex example occurs when a portion of the hierarchy is expanded or contracted. In this case a query is constructed from several items. These items include the node that is to be expanded, a list of currently expanded nodes, and information about the current view context. The query generates an intermediary representation that is a linear list of nodes. For each node the intermediary representation contains:

- Node level — The depth of this node in the display hierarchy.
- Tag Label — A textual label of a node item tag.
- Tag Type — The type of the node item tag.
- Tag Reference — A reference to the location of the tag contents in the file system in case the item must be retrieved.

- Node Expansion state — This is the current display state of this nodes expansion indicator. This can be the obvious 'expanded' or 'contracted' states. Less obvious is the 'none' state which indicates that this node has no expansion indicator.
- Node Label — A textual label used to represent the node item in the outline view.
- Node Reference — A reference to the location of the node item in the file system used when the item must be retrieved and displayed.

This subset of node attributes is sufficient to create a hierarchical outline view that can be used for browsing and viewing items in the archive. An instance of this type of intermediary representation results in a browse view much like the one in Figure 3-1.

The intermediary representation includes only a subset of many node relationships. A query will select node relationships that project a single node hierarchy. Since node relationships can cross databases, multiple databases may be searched during a query. A node that satisfies the query is written to the intermediary representation. When writing the node, the relation that caused the node to be selected is made explicit by removing indirect references. For example, Figure 4-2 shows a stylized representation of two databases in the first two frames. This figure highlights a tree structure of node relations over the other possible structures. If a query were performed on the first database, the intermediary representation would be written as if it had been generated from the merged representation in the third frame. Writing node information and collapsing indirect node references simplifies the generation of a user view from the intermediary representation.

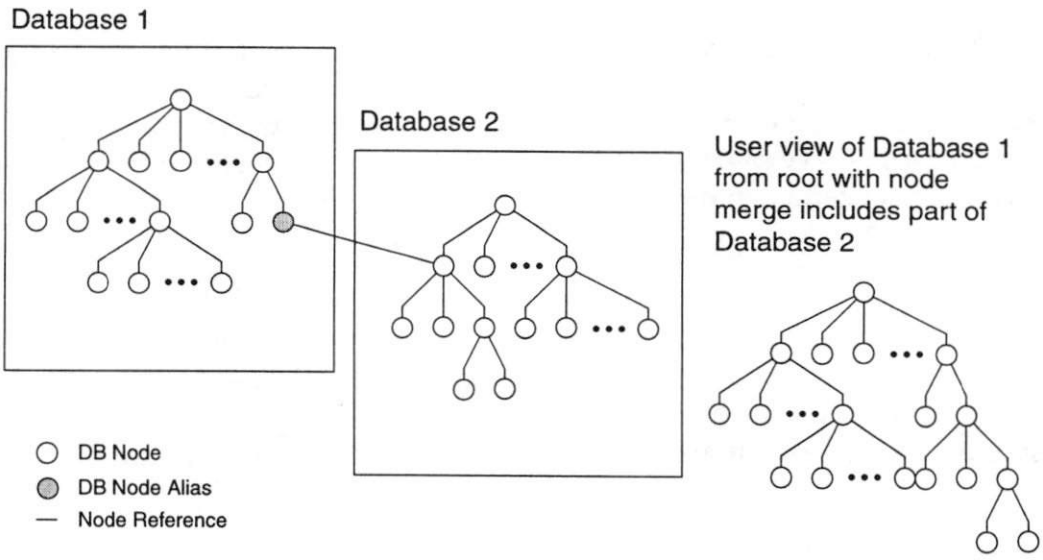

Figure 4-2 Merging views through intermediary representation

The intermediary representation can be used to represent just the structuring information (i.e. hierarchical structure) without the associated archive data. Alternatively, this representation can be used to just present a list of items that are in the archive. That is, the intermediary representation can present structure without data or data without

structure. One example of structure without data is the choice hierarchy used to note placement of new topic nodes. An example choice hierarchy was Figure 3-4 in an earlier section.

The intermediary representation partially bridges the gap between a user's view of the node data and the representation maintained by the system. The intermediary representation includes explicit node data and explicit node relationships, but is not designed for users to view. The presentation backend fills the remainder of this gap by converting the intermediary representation to a user view. The following section describes the presentation backend.

4.5 The Presentation Backend

One fundamental aspect of Collaborative Refinery is the ability to generate different views of the same data which can be edited by any collaborator. Supporting multiple views means that the presentation characteristics for any one view of the data needs to be independent of the data itself. Storing presentation information separate from the data breaks a fundamental assumption of HTML on the Web. The presentation backend attempts to separate the visual and interaction characteristics of the system from the data.

The system hands off the intermediary representation to the presentation backend along with information about the desired type of presentation. The backend uses some or all of the intermediary representation to generate a view based on the users' preferences and the type of application which they are using.

4.5.1 Translating the Intermediary Representation

The backend translates the intermediary representation integrating the intermediary data, markup, and appropriate state information. The intermediary representation is stored and manipulated as complete nodes. The backend processes the intermediary nodes one at a time, reading the node data, merging markup, state and action information, and then dumping the results. This is a very simple streaming translation, in which there is no rewinding or revisiting previously translated nodes. This means that if a node is present twice in a presentation then it must show up twice in the intermediary representation.

The presentation backend has detailed knowledge of the presentation medium. The current system translates the intermediary representation into HTML web pages, integrating the markup and the CGI actions that must be made for each user action.

4.6 Supporting Distillates and Distilling — Catalysts

The prior sections have discussed the major architectural components of Collaborative Refinery. One less obvious component is the extension mechanism that supports the creation and maintenance of distillates. The extension mechanism is known as a *catalyst*. A

catalyst is code which is executed by Collaborative Refinery. A catalyst reads items from the archive and writes a *rough distillate*. A rough distillate is a distillate which will be sent to the user for editing before it is saved by the system.

There are two basic problems in supporting distillates and distilling. The first problem involves how to directly support the creation of a rough distillate from some specified archive contents. The second problem is much deeper. Since there are many potential distillate types, how can the system be designed and implemented to support arbitrary distillate types and new distilling mechanisms? These two questions are answered in the following sections.

4.6.1 Creating Rough Distillates

Currently, Collaborative Refinery supports two rudimentary distillate types. The first is a synopsis type. The synopsis catalyst concatenates message or news items and inserts some markup to emphasize the message contributor, the subject and the message date. The second catalyst type is a table-of-contents type. The table-of-contents catalyst generates a list of hypertext references, one reference to each item being distilled.

The process of creating a rough distillate is the same for all distillate types. The user supplies Collaborative Refinery information through a form. The user indicates the desired distillate type and the child nodes that will serve as the initial content. The distillate type is chosen by selecting a specific catalyst. The initial content is indicated by selecting groups of child nodes that meet some specific criteria. For example, the initial content of a rough distillate may include the contents of several child nodes. One of the child nodes may be an older distillate in need of revision. Additional content may come from child nodes that were recently added to the current node. The system performs a special query which returns a file system pathname and a Collaborative Refinery specific reference. The pathname is used to read the content of a child node and the Collaborative Refinery specific reference is used to create a hypertext link. When the catalyst is run, both the pathname and the system specific reference are provided to the chosen catalyst. The catalyst uses the pathname and the system specific reference to read the node items and write a rough distillate. This rough distillate is then given to the user for editing.

The distillate types supported by Collaborative Refinery can be extended by writing Tcl scripts. Tcl scripts and a Tcl interpreter comprise the catalyst extension mechanism. The Tcl scripts have a specific form for receiving information from Collaborative Refinery and for writing distillates. The implementation of catalysts is covered in the next section.

4.6.2 Extending Distilling — Catalysts

New distillate types can be added to Collaborative Refinery through the use of catalysts. A catalyst is a Tcl script that follows some conventions when interacting with Collaborative Refinery.

These conventions are straight-forward. The catalyst is called with two command line parameters. The first parameter is the name of a Tcl variable file which the system has

composed for the catalyst. The catalyst should source the variable file to have access to query results and various user supplied preferences. The second parameter is the name of a file in which the rough distillate should be stored. This allows Collaborative Refinery to find the rough distillate when the catalyst has completed. Lastly, the standard catalyst convention is for the first three lines of the rough distillate to be the parent node of the new distillate, the file reference of any current distillate attached to that node, and the text title of that distillate. These three lines are important for Collaborative Refinery to reconstruct state both when the catalyst has finished with the rough distillate and when the user has finished editing the rough distillate.

The key problem for someone writing a new distillate is obtaining enough information from Collaborative Refinery to generate a useful rough distillate. One potential solution was to duplicate some of the query mechanism in the catalyst environment. This would allow catalysts to obtain full access to the underlying database. This solution would force a catalyst writer to know more system details, thus requiring a more sophisticated catalyst writer.

Collaborative Refinery's alternative is to perform a larger query in anticipation of many types of access that a catalyst might want to the underlying archive content. The query results are then formatted as a large list of Tcl set commands. The Tcl variables provide a catalyst access to the raw content of each item, access to older distillates, a means of inserting system references into a rough distillate, and access to the user preferences that were active when the particular catalyst was selected. These Tcl variables can be found in Appendix A.

Any Tcl script that uses these variables and standard conventions can be considered a catalyst. There are no restrictions on how the catalyst processes the content items nor on what the catalyst produces as a rough distillate. Naturally, since the rough distillate will be sent to a user for editing, the output should have some meaning to a user.

5. Related Literature and Systems

The Collaborative Refinery brings together and borrows from three research streams. The system relates work on organizing and information management, collaborative authoring, and shared collaborative workspaces. The following sections cover each of these areas in more detail.

5.1 Organizing and Information Management

Categorization and organizing research chronicles the various strategies that individuals use in the day to day use of their personal work space and how the physical location and organizing type (file, folder, pile, stack, event, priority, date, etc.) interact. The stated and implicit goal present in many of these types of studies is to inform the design and implementation of personal digital workspaces, or digital desktops.

Collaborative Refinery's extensive use of references and aliasing derive from Malone's work on individual organizing [9] and Furnas et. al.'s work on differential term usage [5]. Both studies point out the benefits of extensive aliasing. The organizational overview presentation was a specific attempt at dealing with the difference between the organizers intended use of the keywords presented in the browsing view and the seekers understanding of the term.

Berlin, et. al [2] greatly influenced the design space considered for Collaborative Refinery. The four dimensions[2] discussed by Berlin, et. al. are problematic for any system designed to support collaborative classification. Collaborative Refinery attempts to address two of the four dimensions: Purists to Proliferators and Semanticists to Syntacticists. Careful manipulation of a user's view might mitigate some problems with the third dimension, Scruffies to Neatniks. Finally, there is no current way to address style conflicts in the fourth dimension, Savers to Deleters. This fourth dimension points to a fundamental problem that engages issues of organizational policy relative to personal preferences.

5.2 Collaborative Authoring

The range of systems and research that cover collaborative authoring is quite broad. The Computer Supported Cooperative Work (CSCW) and the Hypertext communities have both built systems, studied systems in use, and researched collaborative writing in general. However, Collaborative Refinery implements a different form of collaborative authoring than what has been previously studied.

Collaborative Refinery implements as style of authoring that is much more like abstracting or digesting where the management of sources and collections as well as the creation of text are important. Writing studies like Posner and Baecker [14] are exemplary studies in their attempt to elucidate requirements that support collaborative writing behavior. However the majority of the requirements focus on the creation of new, original material instead of the iterative refinement of text more closely based on specific sources.

---

2. Our view is that the four dimensions are defined by descriptive terms representing opposing behaviors along a continuum. The four dimensions (from Berlin [2]):
   - Purists to Proliferators - This dimension concerns where things go in some set of categories. The fundamental trade-off is between the desire to put everything in a specific well defined place and the recognition that some items may fit into more than one category.
   - Semanticists to Syntacticists - This dimension represents the trade-offs in how people look at categories when doing retrieval. Semanticists strategy is to derive meaning from the category names or titles. Alternatively, syntacticists attempt to develop category titles that contain structural and episodic cues.
   - Scruffies to Neatniks - This dimension represents the trade-offs between having a few broad categories and having many hierarchically organized specific categories.
   - Savers to Deleters - This dimension represents a trade-off between the desire to save everything and the desire to keep only the most relevant items.

Many of the collaborative authoring tools and systems focus on the creation of new text rather than text abstracting and digesting. Systems like GROVE [3], Prep [11], Quilt [8], and ShrEdit [13] are highly effective for geographically close, synchronous collaborations. Collaborative Refinery could incorporate more of the general synchronous editing features for closer collaborations. Collaborative Refinery's focus on providing Web based support results in an implementation closer to the more asynchronous systems like MESSIE [15], Møljner [10] and to some degree PrepNet [12]. Web-based authoring systems, such as Contact [7] and BSCW [1], differ from Collaborative Refinery in that they focus on the coordination and versioning issues more than on the collection management and distilling issues.

Collaborative Refinery might not appear to have a strong tie to hypertext, but the browse view is a general cover for a hypertext system that is presented through the World Wide Web. Research on collaborative hypertext, such as NoteCards [16], SEPIA [6], and Virtual Notebook System (VNS) [4], covers a wide range of issues. Collaborative Refinery inherits the general viewing and editing issues in the above systems. The permission control and collaboration aspects of VNS are not present in Collaborative Refinery. The current assumption in Collaborative Refinery is that everyone is allowed to edit and modify any portion of the hierarchy. SEPIA's emphasis on smooth transitions between loosely coupled asynchronous editing and tightly coupled synchronous editing would be a great asset for the editing of distillates.

The shared editing studies are informative and provide useful design suggestions. However, the primary focus of these studies is collaboration designed to generate new and unique text, a research paper, a newspaper or magazine article, or a book. Studies of collaboration though hypertext provide a slightly different view of a similar set of issues with awareness, coordination, editing and presentation. None of the existing systems have considered hypertext, collections, and editing in the form which the Collaborative Refinery supports. The creation of a distillate is similar to creating a digest, indexing, or abstracting. Likewise, to our knowledge there are no systems to support this behavior or qualitative descriptions of people who perform this behavior.

5.3 Shared Workspaces

Broadly, a shared workspace is any system which supports a view of a collaborative object which can be seen and modified by more than one collaborator. The research literature includes many shared workspaces. The recent development and explosive popularity of the World Wide Web make it a tantalizing target for shared workspaces. The Collaborative Refinery is one of a small number of shared workspaces that are presented through the Web.

Collaborative applications that rely on the World Wide Web use HTTP as the underlying transmission protocol and HTML as a generalized interface. Web based applications present loosely coupled, asynchronous interaction. Asynchronous shared workspaces have difficult problems with simultaneous data access and modification. Therefore many of the Web based workspaces support coarse grained locking of the shared artifacts and

serve to coordinate activity around those artifacts. BSCW [1], GAB [17] and Contact [7] all claim to be examples of shared workspaces that rely on the Web.

Collaborative Refinery is not closely related to any of these workspaces. BSCW and Contact are both heavily oriented around maintaining form-based meta-data about the status of a collaborative artifact. In the case of Contact the artifact is specifically a writing project; with BSCW the artifact is more generalized. GAB projects a browsable hierarchy; however, the way in which individuals modify the workspace is more complex. Collaborative Refinery is unique among these workspaces. In Collaborative Refinery everything is modifiable through the shared workspace; this includes the meta-data, or content structuring, and the content itself.

6. Future Development

Collaborative Refinery addresses several important issues. One interesting issue is the distinctive notion of authoring fostered by Collaborative Refinery. This style of authoring, distilling, is much like creating a digest, an abstract, or indexing. Current research literature considers authoring as a text generation problem in which several people collaborate to generate a new, unique product. The distilling process may generate a unique product, but that product is more clearly based on source material that can be readily identified. As discussed above, there are distinct genres of distillates but the range of different genres is open.

Another issue is that of control over distillates and distilling. Some systems resolve issues of control by specifying roles and allowing users to assume specific roles. It is not clear that there is a strong dichotomy between distillate authors and readers that would allow a role-based solution. The skills and expertise of people vary and likewise their needs to alternately read and author distillates will vary.

Control over distillate readership presents another, more subtle set of access issues. It is conceivable that two or more distillates could be derived from exactly the same contents. In this case, which distillate should be presented to which users? One distillate may contain extremely sensitive information that requires certain system based or socially based privilege for access. Alternatively, different distillates from the same sources may be targeted for users of different skill levels (e.g. novices or experts). Resolving these subtle access issues for different distillates is an open problem.

Other distilling issues to be addressed concern how to identify distilling scope, how to effectively identify stale distillates, and how to effectively assist the social aspects to each user's satisfaction. One major research focus must be support for distilling and distillates.

The Collaborative Refinery presents an integration of collaborative information management with collaborative authoring. Collections and collection management have been missing from the collaborative writing literature. Collaborative Refinery supports collection management by multiple collaborators with multiple hierarchical views, and topic

aliasing. This strategy mitigates, but does not solve, several of the conflicts discussed in [2]. One area of potential work will be to provide better support for the resolution or mutual cohabitation among conflicting organizational styles.

7. Summary

The Collaborative Refinery represents an alternative method of identifying interesting and useful items in an exploding morass of information. The basic approach is to leverage the collaborative work of interested, motivated individuals, and experts. Collaborative Refinery supports basic behaviors, collecting, culling, organizing, and distilling, in a shared workspace which enable this potential solution. Supporting these four general behaviors integrates work in information and collection management with collaborative authoring.

8. References

1. Bentley, R., Horstmann, K. Sikkel and J. Trevor. Supporting Collaborative Information Sharing with the World Wide Web: The BSCW Shared Workspace System. *Proceedings of WWW'94*, 1994:

2. Berlin, L. M., R. Jeffries, V. O'Day, A. Paepcke and C. Wharton. Where Did You Put It? Issues in the Design and Use of a Group Memory. *Proceedings of InterCHI'93*, 1993: 23 - 30.

3. Ellis, C. A., S. J. Gibbs and G. L. Rein. Groupware: Some Issues and Experiences. 1991, 34(1):

4. Fowler, J., D. G. Baker, R. Dargahi, V. Kouramajian, H. Gilson, K. B. Long, C. Petermann and G. A. Gorry. Experience with the Virtual Notebook System: Abstraction in Hypertext. *Proceedings of CSCW '94*, 1994: 133 - 143.

5. Furnas, G. W., T. K. Landauer, L. M. Gomez and S. T. Dumais. The Vocabulary Problem in Human-System Communication. 1987, 30(11): 964 - 971.

6. Haake, J. M. and B. Wilson. Supporting Collaborative Writing of Hyperdocuments in SEPIA. *Proceedings of CSCW '92*, 1992: 138 - 146.

7. Kirby, A. and T. Rodden. Contact: Support for Distributed Cooperative Writing. *Proceedings of ECSCW'95*, 1995: 101 - 116.

8. Leland, M. D. P., R. S. Fish and R. E. Kraut. Collaborative Document Production Using Quilt. *Proceedings of CSCW'88*, 1988: 206 - 215.

9. Malone, T. W. How Do People Organize Their Desks? Implications for the Design of Office Information Systems. 1983, 1(1): 99 - 112.

10. Minor, S. and B. Magnusson. A Model for Semi-(a)Synchronous Collaborative Editing. *Proceedings of ECSCW '93*, 1993: 219 - 231.

11. Neuwirth, C. M., D. S. Kaufer, R. Chandhok and J. H. Morris. Issues in the Design of Computer Support fo Co-authoring and Commenting. *Proceedings of CSCW '90*, 1990:

12. Neuwirth, C. M., D. S. Kaufer, R. Chandhok and J. H. Morris. Computer Support for Distributed Collaborative Writing: Defining Parameters of Interaction. *Proceedings of CSCW '94*, 1994: 145 - 152.

13. Olson, J. S., G. M. Olson, L. A. Mack and P. Wellner. Concurrent Editing: The Group's Interface. *Proceedings of IFIP 3rd International Conference on Human-Computer Interaction (INTERACT'90)*, 1990: 835 - 840.

14. Posner, I. R. and R. M. Baecker. How People Write Together. *Proceedings of Twenty-fifth International Conference on the System Sciences*, 1992:

15. Sasse, M. A., M. J. Handley and S. C. Chuang. Support for Collaborative Authoring via Electronic Mail: The MESSIE Environment. *Proceedings of ECSCW '93*, 1993: 249 - 264.

16. Trigg, R. H., L. A. Suchman and F. G. Halasz. Supporting Collaboration in NoteCards. *Proceedings of CSCW'86*, 1986: 153 - 162.

17. Wittenburg, K., D. Das, W. Hill and L. Stead. Group Asynchronous Browsing on the World Wide Web. *Proceedings of WWW'95*, 1995:

Appendix A — Catalyst Variables

This is a list of exported Tcl variables that are supplied to a Collaborative Refinery catalyst. The variables are exported to the catalyst through a variable file which Collaborative Refinery creates just before the catalyst is executed.

- DBRootPath – The file system path to the current active database.
- CurrentDB – The current active database in the root path.
- DistRootPath – The file system path to the distillates.
- ItemRootPath – The file system path to the individual archive items.
- DisplayURLPrefix - A URL prefix used by the system to retrieve an item.
- ParentNode – The system specific parent node for the new, prospective, distillate.
- DistFileRef – The file system reference of the old distillate stored at the ParentNode node, if any. This should be used in conjunction with DistRootPath for access to the actual distillate file.
- DistDBRef – The database reference to the old distillate stored at the ParentNode, if any.
- DistTitle – The title of the old distillate at ParentNode, if any.
- NewItemFileRef – An array of file names. A catalyst can use these along with ItemRootPath to read each of the content items that should be distilled.
- NewItemDBRef – An array of database references, used by the system in conjunction with the DisplayURLPrefix to retrieve and display the indicated item.
- NewItemTitle – An array of titles, one for each item that should be distilled.
- NewItemCount – The number of items in the NewItem arrays.
- UseItems – This indicates the users preference for which items, 'new', 'old', 'all', or 'none' should be used in the creation of the new rough distillate. The 'new' value is the only one which currently makes sense here, because the system does not keep track of the times when content items are added to nodes.
- UseDistillate – This flag indicates the users preference to include the old distillate, if any, in the generation of the new rough distillate or not. If the value of UseDistillate is 1 then the old distillate should be used.