Information Access in Complex, Poorly Structured Information Spaces

Gerhard Fischer & Curt Stevens

CU-CS-461-90 February 1990

Department of Computer Science
Campus Box 430
University of Colorado @ Boulder
Boulder, Colorado 80309-430

UNIVERSITY OF COLORADO AT BOULDER TECHNICAL REPORT CU-CS-461-90, FEBRUARY 1990

Information Access in Complex, Poorly Structured Information Spaces

Gerhard Fischer & Curt Stevens
Department of Computer Science and Institute of Cognitive Science
University of Colorado, Boulder

Abstract

This research extends our previous efforts on information access (centered around the HELGON system) by choosing an information space which is *less structured* than the information store represented by HELGON. The domain studied is the News system available on computers. The amount of information distributed and made available through News creates a serious information overload.

The *conceptual framework* behind this research effort explores (a) the relationship between situation models and system model (specifically the question how the system model can be restructured over time to get closer to an individual user's situation model) (b) the role of structure in dealing with large information spaces (specifically: where does it come from, who is willing to generate it, whose structure is it?).

The *innovative system building effort* (instantiating the conceptual framework as well as extending it) is centered around the INFOSCOPE system which focuses on the following issues: (a) it allows users to construct *virtual newsgroups* to reduce the size of the information space (b) it supports the restructuring of the information space at read time according to individual semantics (c) it makes no assumption that senders of a message do any extensive structuring and and allows users to impose their own semantics (d) it incorporates agents which assist users (based on information accumulated in a user model) in suggesting better ways to deal with the information space and in restructuring it.

The approach taken by INFOSCOPE differs from other approaches which require more up-front structuring. We believe that INFOSCOPE has major advantages in realistic working environments where people are unwilling to spend time and effort on tasks which are of no benefit to them. INFOSCOPE is an operational system and will be used as a general tool in our research group -- providing us with the opportunity to conduct longitudinal studies in a realistic setting.

Acknowledgements

The authors wish to thank the other members of the ARI project, especially Robert Bernard, Evelyn Ferstl, Peter Foltz, Scott Henninger, Walter Kintsch, and David Redmiles with whom we shared many discussions about the ideas and the systems discussed in this paper. The research was supported by Grant No. MDA903-86-CO143 from the Army Research Institute

Table of Contents

List of Figures

Information Access in Complex, Poorly Structured Information Spaces

Gerhard Fischer & Curt Stevens
Department of Computer Science and Institute of Cognitive Science
University of Colorado, Boulder

1. Introduction

As local area networks give way to wide area networks, and those in turn give way to global networks, the amount of electronic information available to people on their desktops is becoming staggering. One global network which amply represents this problem is the Internet, specifically in its distribution of USENET News. Because of the manner in which new News is accepted into the distribution system, users find themselves constantly confronted by problems associated with large complex information spaces. When users wish to distribute news (messages) through USENET, they must first choose a *newsgroup* into which the message is classified. Readers then find the message in that newsgroup. However, the semantic classifications (vocabulary etc.) of the sender are not necessarily the same as those of the thousands of readers who might possibly read this message. The sender must classify it by posting to a specific newsgroup or set of newsgroups and therefore a somewhat general classification hierarchy has been established in USENET News (see Figure 4.1). The task of finding relevant information is one which requires the perusal of many irrelevant messages. An empirical study which we carried out demonstrates that users only subscribe to a fraction of the potentially interesting newsgroups for this very reason. Since messages expire at regular intervals the problem is even worse. The transitory nature of the data makes it crucial that readers can find relevant information in a timely manner. The size of the information space can frustrate users' ability to effectively utilize the potential of the available information.

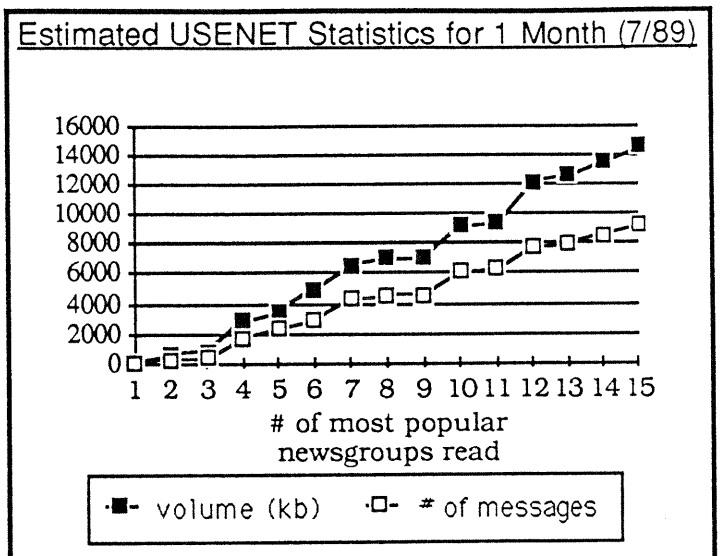
What follows is a discussion of the INFOSCOPE project which explores methods of helping alleviate problems associated with the large amount of information found within the context of USENET News. A conceptual framework for the project will be developed. The implementation of INFOSCOPE encapsulates most of these ideas in a fully functional news news access system.

2. Empirical Analysis

2.1 Information Overload

Information overload refers to the problems created by the huge amount of information that is available to us as information consumers [Reddy 1983]. In addition to over 100 million volumes in the Library of Congress and thousands of daily, weekly and monthly publications, advances in microprocessor technology, communications networks, and computer systems will be unlocking access to new types of intelligent networks and information previously unavailable to computer users [Schwartz 1989]. With all of this information available, the problem of finding interesting information can be compared to that of *finding a needle in a haystack*. However, it is clear that no individual actually wants to read all of this information, even if it were possible. People are interested in learning about or investigating only certain areas of personal interest. Uninteresting information (that which people do not wish to read) only gets in the way of those tasks. For example, we often refer to

unsolicited mail as *junk mail*. As methods of proliferating information advance, so does the amount of irrelevant information we must wade through in order to find the answers we need.



2.1 USENET News Traffic

With the advent of global networks like USENET, the amount of available information can be staggering. Reading just a few of the most popular newsgroups requires browsing megabytes of data each month!

In the INFOSCOPE project we are investigating a particular network information system, USENET News. While this system represents only a small fraction of the information mentioned above, it still represents a large enough information space to present its users with all of the problems associated with information overload. Data shows that even reading a few of the more popular newsgroups requires the perusal of megabytes of information each month (see Figure 2.1). With over 400 newsgroups to choose from even the most prolific readers cannot read more than a small percentage. Our empirical evaluation showed that users are interested in reading about more topics, but the large volume of information makes that impossible. Many of these responses were from readers who subscribe to only two or three newsgroups!

Computers serve well as a medium for making more and more information available to their users. The critical resource is not the available information but the human attention necessary to utilize that information [Simon 1983]. Moreover, computer systems should be designed so that users can focus their individual attention on whatever information they conclude is useful to them, regardless of the criteria they use to arrive at those conclusions.

2.2 The Classification Hierarchy behind News

The USENET News information space is organized around a hierarchy of newsgroups. Each node in the hierarchy represents either a specialization of the hierarchy or a newsgroup that contains messages. An example of several newsgroup names can be seen in Figure 2.2. A path through the hierarchy is used to identify a particular node and a 'dot' separates each element of the path. General categories in the hierarchy range in topic from *recreational* (rec.), to *computer* (comp.), to local categories like *Colorado* (co.) among others. Since classifications are very general the software allows users to *cross-post* messages to several newsgroups at one time. For example, a message about problems formatting this paper could potentially be posted to the newsgroups comp.sys.mac, comp.desktop, and comp.windows.

In addition to this problem, the vocabulary problem manifests itself quite prominently in the use of the newsgroup hierarchy. When users post to a particular newsgroup they are necessarily imposing their own semantic interpretation on the message by placing it in a specific part of the newsgroup hierarchy. The problem is that not everyone understands each classification in the hierarchy to contain the same set of message topics. One approach to reducing this problem is to allow the hierarchy to evolve. This has been implemented, but the process is slow and requires the agreement of significant numbers of people. For this reason changes often get proposed several times before enough users are convinced or interested enough to cast

votes in favor. This problem is even worse for those individuals who do not agree with accepted changes. They are still forced to access information through other people's conceptualization of its structure.

2.3 The User Interface to News

Several interfaces to the news information space have been implemented. Most of them have been text based and designed to operate on conventional tty type terminals. The most common of these interfaces is called RN. This system presents users with a succession of text lines containing the names of newsgroups and the number of unread messages contained in that group (see Figure 2.2). An init file is kept by the software to track the newsgroups which are subscribed to by the user (a user subscribes by requesting messages from that group). This init file also determines the order in which newsgroups are presented but the system comes with no tools for managing this file. To override the default order users can either edit the init file or ask the software to go directly to a specified group. While this type of an interface might be fine for a user who is both experienced with the software and familiar with the organization of the information space, it does not illustrate this structure in an effective manner.

2.4 Support for Conversation

Another aspect of the RN software that confounds many novice users is the lack of structure imposed on logical collections of messages. Conversations between people are often more than "one-shot" dialogs. When several users post a response to a particular message, those messages may each spawn sub-conversations themselves. These sub-conversations are often called *threads*. RN displays messages in the chronological order in which they arrive at an individual's computer. Although the system provides a command for scanning to the next message in the overall conversation, the structure of individual threads is lost in the chronological ordering. In fact, due to the way messages are propagated throughout the network, situations can arise where a response to a message is displayed prior to the original message.

The RN Interface to Newsgroups

```
*** 1 unread article in alt.hypertext--read now? [ynq]
*** 13 unread articles in alt.sources--read now? [ynq]
*** 1 unread article in boulder.general--read now? [ynq]
*** 2 unread articles in co.general--read now? [ynq]
*** 23 unread articles in comp.ai--read now? [ynq]
*** 14 unread articles in comp.binaries.mac--read now? [ynq]
*** 9 unread articles in comp.cog-eng--read now? [ynq]
*** 99 unread articles in comp.lang.lisp--read now? [ynq]
*** 1 unread article in comp.mail.headers--read now? [ynq]
*** 6 unread articles in comp.newprod--read now? [ynq]
*** 11 unread articles in comp.sources.games--read now? [ynq]
*** 30 unread articles in comp.sources.misc--read now? [ynq]
*** 1 unread article in comp.sys.mac.digest--read now? [ynq]
*** 289 unread articles in comp.sys.mac--read now? [ynq]
```

2.2 Text Based Newsgroup Interfaces

Text based interfaces do not display the relationships between newsgroups in the information space. Browsing in this type on environment can be an arduous task.

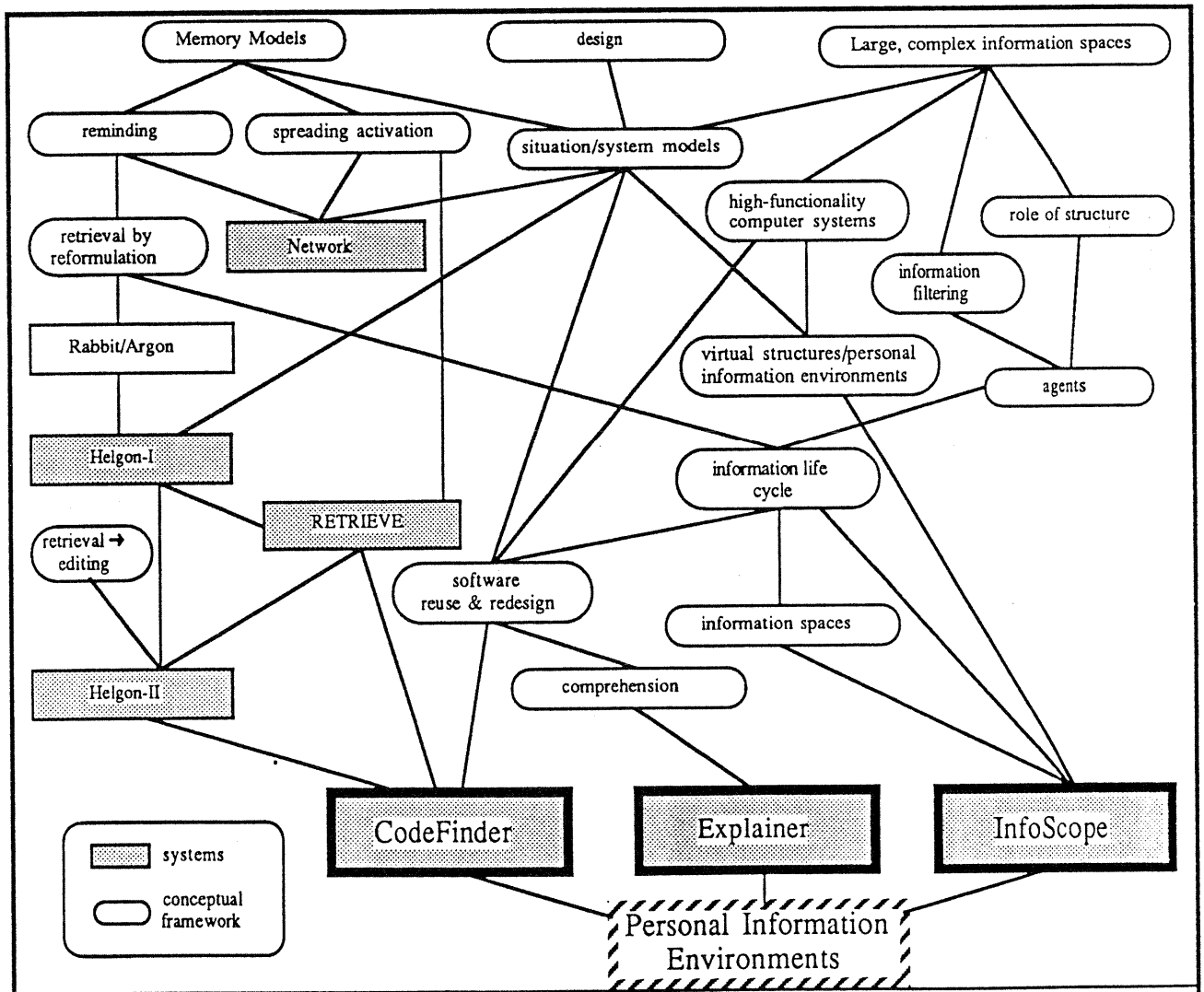
3. Conceptual Framework

The INFOSCOPE project is not an isolated attempt to solve problems in the information access domain. As illustrated in Figure 3.1, it is only one part of an overall attempt to develop the technologies necessary for the realization of personalized information environments. These adaptable systems are crucial in dealing with information overload [Reddy 1983].

In order for these systems to be useable, they must include some amount of intelligence which can be used to shift much of the information processing burden from the individual to the system. An integrated system must help deal with information throughout its lifetime, rather than just at one point in it lifecycle as isolated systems tend to do.

3.1 An Information Lifecycle

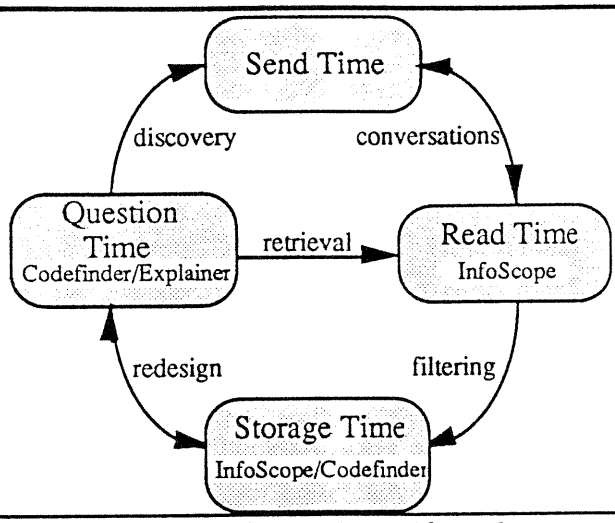
Like many artifacts in our lives, the messages we send and receive can be viewed as having their own lifecycle. When developing systems to help with messages we must recognize the processes which a message goes through in travelling from step to step in its lifecycle. For example, as illustrated in Figure 3.2, when related messages traverse the arc between *send time* (when messages are created) and *read time* (when messages are read by someone) a *conversation* is the result. However, only some messages are interesting over the long term, and those messages traverse the arc between read time and *storage time* (when messages are saved for later retrieval). Saving messages is one of the operations INFOSCOPE uses to help determine which message topics are interesting. A saved message clearly is interesting to the person who saved it.



3.1 Project Overview

The INFOSCOPE project is only part of a larger search for the technologies needed to drive personal information environments [Fischer, Henninger, Redmiles 1990] By creating an integrated environment for all of the tools, we are developing systems which can deal with messages throughout their useful life.

Once a message is stored for future reference, this is not the end of its lifecycle. When people have a question they can access their personal information store to find relevant information. This process of taking a message from *question time* to read time is called *retrieval*, and is addressed by the HELGON system [Fischer, Nieper-Lemke 1989]. When a user finds a relevant piece of information it can also be modified. This is often the case when the content of particular messages contains artifacts like source code. The process of modification and restorage into the personal environment as a new object can be refered to as *reuse and redesign* [Fischer, Henninger, Redmiles 1990].



3.2 An Information Lifecycle

There are 4 basic steps in the life of a message. Within each step there are several tasks and decisions to be made, and a system(s) to help with them. The arcs in this diagram represent the processes involved in transferring a message from one step to the next.

Finally, there is the situation where the users have a question, but have not previously stored any relevant information in their personal environments. In this case users engage in a process of *discovery* [Schwartz 1989] in which a new message is posted to the outside world as a request for augmentation of one's own environment. Since the process of discovery involves interaction with many agents (in this case the other users of USENET News), it is highly unlikely that any two identical queries will result in the same set of answers. This makes the process of discovery quite different from that of retrieval. The same query to the same database by several individuals should always yield the same results in a retrieval situation.

Inside the Four Steps In Information Processing – The whole story of the information lifecycle cannot be told through the arcs which transfer messages from one step to another. The four distinct processing steps each represent several actions and issues with which users must deal (see Figure 3.3). The first of these is *send time* processing. Send time processing is the work users must do in order to send a message. This processing currently ranges from simply specifying a destination and subject (as in UNIX e-mail), to coarse grained categorization (as in USENET newsgroups), to fine grained structuring of the semantics of the message content (as in the LENS system discussed in section 3.3).

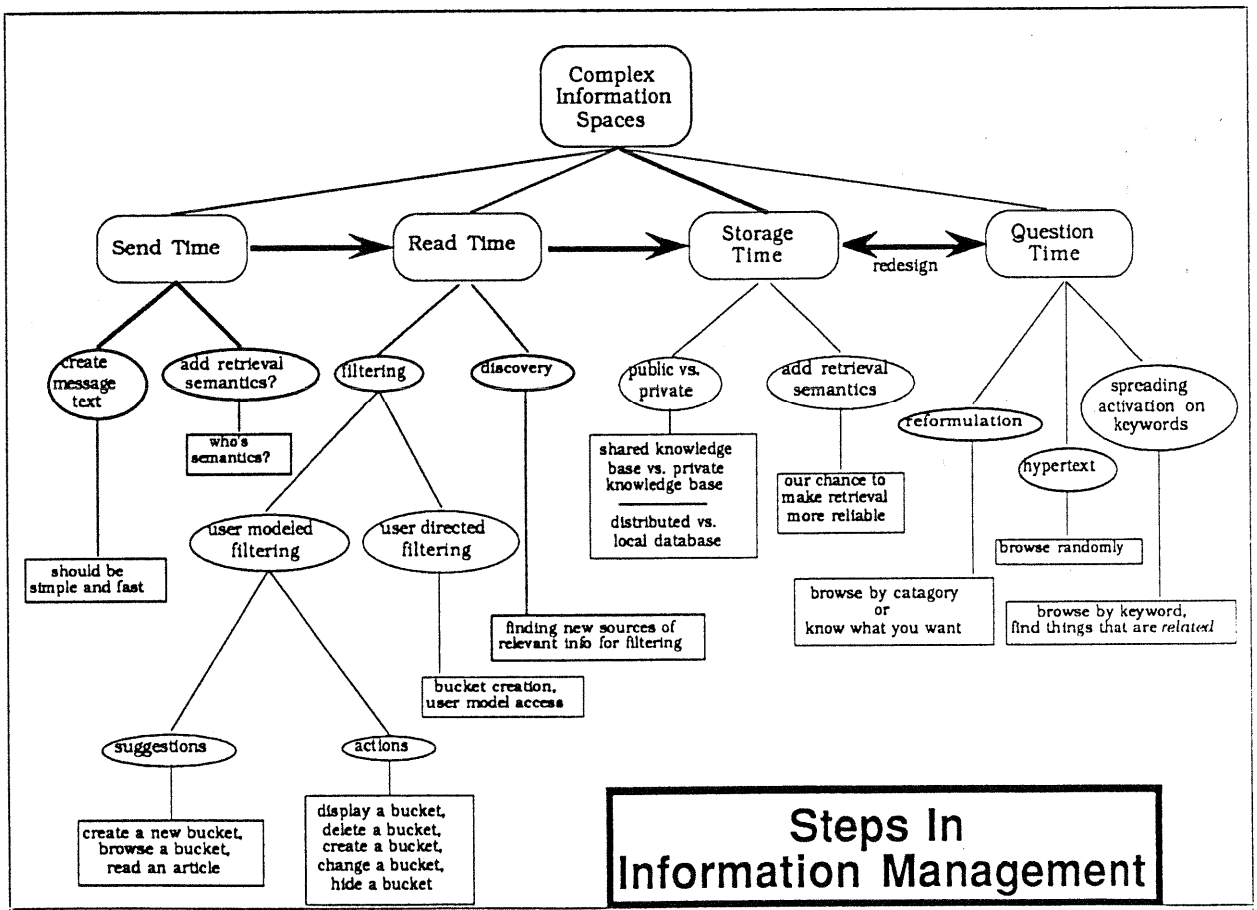
After send time processing the next step is *read time* processing. Read time processing is the work users must do in order to find and read the interesting information from within the complex information space. This involves a range of activities from simply browsing the information space (like the UNIX RN news reader), to writing specific rules to filter out information which already includes sophisticated semantics (like the LENS system).

Once a message has been read, a decision must be made as to whether or not the message should be saved for future retrieval. If the decision is for saving the information, the user then has an additional opportunity to add retrieval semantics to the message. We call this storage time processing. This currently ranges from simply saving all messages to a flat file or mbox (like many people use UNIX e-mail), to adding coarse grained semantics by saving to a folder structure (like some people use UNIX e-mail), to utilizing fine grained semantics and saving to a public area which others can peruse using rules (like the LENS system).

Finally, there is the processing which must take place when the information is needed at a later time. This is the information retrieval problem as discussed in the literature and we call this question time processing. Question time processing currently ranges from simply searching for text strings, to utilizing coarse grained semantics (like the UNIX e-mail folders), to utilizing fine grained semantics in the information (like the HELGON or LENS systems).

3.2 Situation and System Models

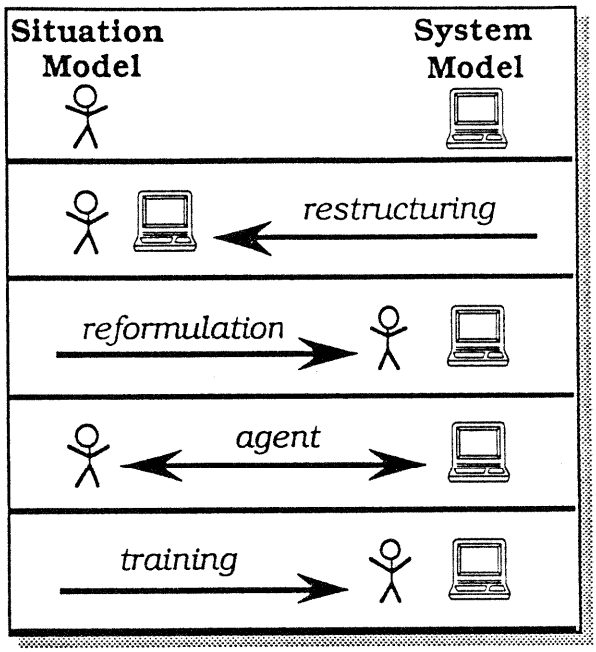
In addition to the information lifecycle, an analysis of the situation and system models [Dijk, Kintsch 1983] plays an important role in the conceptual framework of the INFOSCOPE project. Users state their desires in the terms of their current task, context or view of the world (the *situation model*), while a computer system presents them with an information space organized around someone else's view of the world (the *system model*). For example, a user is interested in the language CLOS (common-lisp object system), but must read each of the newsgroups comp.lang.lisp and cu.cs.commonloops to find such information. In our research group we have investigated various methods for bridging the gap between situation and system models. Several of the approaches offer the user help in translating their situation model into the appropriate system model (see Figure 3.4). Another approach (taken by virtually all commercial software products) is training users to express themselves directly in the appropriate system model. This requires users to read and



3.3 The Life of A Message

A message is created at *send time* and is either addressed to a set of individuals or posted to some sort of bulletin board system. At *read time*, people looking for information or answering e-mail must decide which of the newsgroups and/or messages are worth reading. Sometimes, when a message is of particular interest it is stored by the user for later reference. At this point (*storage time*) the user adds some sort of semantics to the message, ranging from simply specifying a filename to adding keywords and classifications. Finally, stored messages are retrieved at *question time* when a question triggers their need.

comprehend long and complicated manuals. The drawback of these approaches, is that users end up expressing themselves in a system model. At some point the user is required to map from their personal situation model to a more global system model, a task which consumes valuable cognitive resources. An additional approach (investigated through INFOSCOPE) is to develop systems which learn about users in order to allow the expression of structure in the user's own situation model. The system model presented by INFOSCOPE is refined over time, through the development of a persistent user model, until its system model closely matches the situation model of the user. Eventually, the user is no longer required to map into a system model.



3.4 Different Approaches in Relating Situation and System Models

- The first row illustrates the situation where there is no support for bridging the gap.
- The second row shows the approach in which a new system model is constructed that is closer to an individual's situation model and hence easier to understand. This approach is pursued with the INFOSCOPE system.
- The third row illustrates the possibility of making the system model more transparent, allowing users to express their situation model incrementally within the system model. This approach is investigated in the HELGON and CODEFINDER systems.
- The fourth row shows how an agent can help translate a query from the situation into the system model, an approach whose theoretical foundations are explored in the NETWORK system.
- The last row illustrates the training approach, where users are trained to express themselves in the system model.

The gulfs of execution and evaluation – The notions of situation and system models do not stand alone in their attempt to explain the difficulties encountered by users of computer systems (or any *designed* artifact). A similar perspective on this problem has been discussed by Norman and is referred to as the gulfs of execution and evaluation [Norman 1986]. In this analysis, the gap between situation and system models is divided into two distinct unidirectional gaps. The first of these corresponds to the problems of translating a situation model into a system model. It is called the gulf of execution because it involves the translations necessary to go from one's goals to the physical system being used to execute those goals. The gulf of execution is bridged by making the system match the user's view of the domain as closely as possible.

The second directional gap is the gulf of evaluation. This gulf corresponds to the problems of translating from a system model back into one's personal situation model. In terms of this model it represents the mapping from the physical system which has completed some portion of the desired task to the terms and expressions of the goal as viewed by the user. This gulf often causes problems since the representations which constitute an answer on the computer screen are not readily interpretable by users. The gulf of evaluation is bridged by ensuring that the output of the system presents an understandable conceptual model to the user.

The External-Internal Task Mapping Analysis – Another analysis of the problems users encounter when attempting to carry out some task on a computer system is Moran's external-internal task mapping analysis [Moran 1983]. In this model the

computer system is viewed as "a conceptual world into itself — it represents conceptual objects and carries out conceptual actions on those objects." Again, this analysis points out that the major problem is that computer systems are designed, and in the process of design certain conceptual mismatches arise between the system's internal representations and the external reality of tasks and their specifications in the domain of interest to the computer user. The EITM model is a design model which stresses the structure of the external task as opposed to its requirements. For this reason one of the postulates of this model is that systems must provide users a method for getting their task "into the system" by reformulating the specification of the task until is is understood by the system. The HELGON system takes this structured approach to query specification (see Figure 3.4)

3.3 The Role of Structure

Another important aspect of this conceptual framework is the role which structure plays within it. In the INFOSCOPE system there are two types of structure to discuss. The first of these is the a priori structure that defines the USENET hierarchy of newsgroups. The second is the *virtual* structure which INFOSCOPE allows users to create. In addition to these two types of structure, there are also several aspects of each type to discuss.

When discussing systems such as HELGON or INFOSCOPE, which utilize a rich structure, it is important to ask, "*where does this structure come from?*" In the cases of HELGON and the predefined USENET hierarchy the structure is defined by system administrators. Users of the two systems may take part in the addition of more structure, but the original conceptual layout of the structure remains throughout its lifetime. This situation is a major source of the vocabulary problem in utilizing the News hierarchy.

Another important characteristic of structure is that the process of modification requires cognitive effort. This means that there are associated costs and benefits which people can analyze in order to decide if performing a given task is *worth the effort.* Research in computer supported cooperative work has shown that people are reluctant to perform tasks for which there is little perceived benefit to the performer [Grudin 1989]. This means that in order to get users to modify structure, they must feel that this modification will be of benefit to them in the future. If this is not the case, the result is no modification or suboptimal modification of the structure in question.

Of course there is no point in having a rich structure if it is not used to browse the information space it represents. In order to do this, users must have a clear conceptual understanding of the structure as it is represented to them by a computer system. This relates once again to the vocabulary problem discussed earlier. Because the News hierarchy is predefined, there is an a priori mapping of semantics to newsgroup names which may or may not match the semantic interpretations of these mappings by any individual user. Without a clear understanding, the structure can actually do more harm than good. An empirical analysis carried out on users of the HELGON system indicates that users were confused when the relationships between nodes were not clear due to the vocabulary that was used. This made the assigned retrieval task more difficult.

The Information Lens – This system [Malone, Grant, Turbak 1986] was designed to let users add structure to messages at send time (see Figure 3.3). This means that the sender of a message selects a template from hierarchy of templates. This template represents a particular message type which defines the fields needed for a message of that type. While this idea works great in a small environment of users where standards and expectations can be imposed, it may not work well in a global environment where people just want to get work done [Carroll, Rosson 1987]. The reasons for this relate to the cost/benefit ratio as perceived by the sender of messages. Since the sender is not the reader, and since adding structure helps the reader find the message easier, there is no direct benefit to the sender from adding

structure at send time. A typical sender merely wishes to write the text of a message and send it to a particular person or group. INFOSCOPE is based upon the assumption that senders will not expend the extra cognitive effort necessary to carefully classify messages or fill out specific templates at send time. This is done by allowing users to restructure the information space according to individual semantics at read time. Since this restructuring makes access at read time easier for the reader there is direct benefit perceived and therefore more incentive to carry out the structuring process.

Helgon – The HELGON and CODEFINDER systems (see Figure 3.1) use retrieval by reformulation and spreading activation techniques to help users retrieve information from an information space without necessarily knowing exactly what it is they are looking for. However, in order for retrieval by reformulation to proceed, the information space must first be highly structured. Once again we must ask where this structure comes from. In this case HELGON is much like a traditional database in that its underlying structure is created by a database administrator. A subsequent version of this system has added the ability to edit the contents of retrieved items, but the problems arising from a priori structure still remain.

Also, it is important to understand what steps in the information lifecycle each system addresses. The INFORMATION LENS system concentrates its abilities across all steps in the lifecycle, but in so doing relies on having users structure messages at send time. In addition, this approach ignores the fact that many external information sources lack any sophisticated structure (eg., AP Newswire) and therefore systems must be flexible enough to handle unstructured information. This means being capable of presenting a conceptually coherent representation of the information space in question.

The HELGON system has capabilities in the storage time and question time domains (see Figure 3.2). At question time it allows users structured access to an information space. If the structure presented is misunderstood by the user, or if the user is not sure of the precise goal, a reformulation process can be carried out in order to map the query into the system model. The CODEFINDER system adds the ability to spread activation across a network of nodes to retrieve items *related* to a selected item. At storage time these systems allow the user to store modified versions of recovered artifacts, perhaps changing the semantics of the artifact in that process.

The INFOSCOPE system concentrates on the domains of read time and storage time. At read time the system allows users to browse the information space and read any messages found. Users may also choose to modify the structures that represent the information space on the display.

As the previous discussions indicate, information access systems are faced with the dilemma that:

- users need rich structure in information spaces so that a clear understanding of the conceptual structure is maintained
- many sources of information do not exhibit rich structure
- a priori structure is insufficient due to mismatches between internal and external interpretations of structure
- users are willing to add structure themselves only if there is a perceived benefit from adding it

3.4 Personalization

This dilemma leads to a need for personalization of the information space. The methods for doing this however must maintain that perceived benefit. The key to doing this lies in allowing users to add structure at read time as opposed to requiring

them to add it at send time. In this way users benefit directly from the effort expended in modifying their personal view of the information space.

There are other indications of the need for personalization. Not only are situation models often quite different than the system models in which they must be expressed, but these situation models can be quite different for individual users of a system. Since INFOSCOPE must allow the expression of goals in a user's own situation model, it seems that a single system model may not be adequate. What we need are many system models which closely match individual situation models. However, anticipation of each and every possible situation model is impossible since situation models are highly dynamic in nature. After all, situations are changing constantly. Therefore, rather than force the designer to attempt this impossible design task, it may be better to allow the user to define his own system model which is directly based upon his own situation models. In INFOSCOPE this is accomplished through the definition of *virtual newsgroups*. These virtual structures are defined by the system and user for the purpose of personalizing the displayed system model (eg., the current hierarchy). Using this facility, the users discussed earlier who needed to read about CLOS but instead had to read commonloops and lisp newsgroups can define a new newsgroup called comp.lang.lisp.clos (or whatever else suites their taste in newsgroup names). Since the new group only exists within the personal environment of the defining user, and not in the public version of the USENET structure, we call this new a virtual structure.

Baskets – As discussed before, the newsgroups in USENET News are arranged in a general classification hierarchy. In the INFOSCOPE interface not only are newsgroups represented, but so are the entire virtual and predefined hierarchical structures. As demonstrated by experiments with the HELGON system [Fischer, Nieper-Lemke 1989], representing the intermediate nodes of the hierarchy helps make the system model more transparent and therefore easier for the user to understand. As a result of this, not all nodes in the screen display represent newsgroups. For this reason nodes in the hierarchy are referred to as *baskets*. Some baskets contain messages and others contain additional baskets which refine the hierarchy. In this way, baskets are similar to UNIX files (which also serve as directories) or MACINTOSH folders/documents. INFOSCOPE baskets which contain messages can be thought of as newsgroups.

Unfortunately, this kind of system model personalization, like any structure modification task, is very difficult. What we need is a suite of tools which make the task easier to accomplish. To do this in a computer system we need to model the user based on certain concepts of how users act, and we need to develop methods by which these models can be analyzed for the purpose of structure creation and modification (the essence of evolution). In INFOSCOPE, we call these methods agents.

In addition to the complicated task of changing the input language of a system through the evolution of its system model, we must pay close attention to the system's output language as well. In order for the situation/system model gap to be completely spanned, the personalization of the input language must become apparent in the interfaces output language. In the INFOSCOPE system this means creating different representations for the virtual structures which users and agents create and modify over time. INFOSCOPE uses a modified version of the TRISTAN graph display system to accomplish this new representation [Nieper 1985].

4. Infoscope

The INFOSCOPE project is not just a theoretical investigation. It includes a system implementation which embodies many of the principles discussed earlier. This system currently contains four tools (or modules) which allow users to manipulate and personalize the information environment as distributed through USENET News. The *browser tool* represents the environment on the screen so that users can interact

with it in a direct manipulation fashion [Hutchins, Hollan, Norman 1986]. The *message reading tool* lets users read previously posted messages and the *message posting tool* lets users respond to those messages. The fourth tool is the *basket filter tool* which supplies the user with the means to personalize the environment through the extension of its structure. Finally, due to the complicated nature of performing the task which the basket filter tool supports, a fifth module, the *agent tool*, is being designed and implemented. This tool will involve the user in interactions with agents which are designed to ease the burden of extending the environment's structure (see section 4.4).

4.1 Information Structures Interpreted by Infoscope

Specific sources of knowledge to INFOSCOPE include the newsrc file (keeps track of what messages have been seen), information about existing newsgroups which is available through NNTP (Network News Transfer Protocol: contains lists of active newsgroups, new newsgroups, available messages and their id numbers), the filters which are defined over time (by individuals, and patterns recognized over groups of individuals), the user model which contains statistics and interest profiles for individual users and newsgroups, as well as the heuristics contained within the agents themselves.

The first and most familiar news file to current readers will be the newsrc file. This file keeps track of which messages are currently available in the information space (eg., which messages have not yet expired) and which of those available messages have already been read. Another piece of information kept in this file is data about which groups are subscribed to by individual users. This information is initially used by the system as an indication of the existing newsgroups which contain interesting information for the user. The format of this file is not changed by INFOSCOPE so that users may continue to read news from their normal home machines when that is convenient. In fact, INFOSCOPE will read newsrc data across a local area network so that the same newsrc file is used by INFOSCOPE and programs like RN.

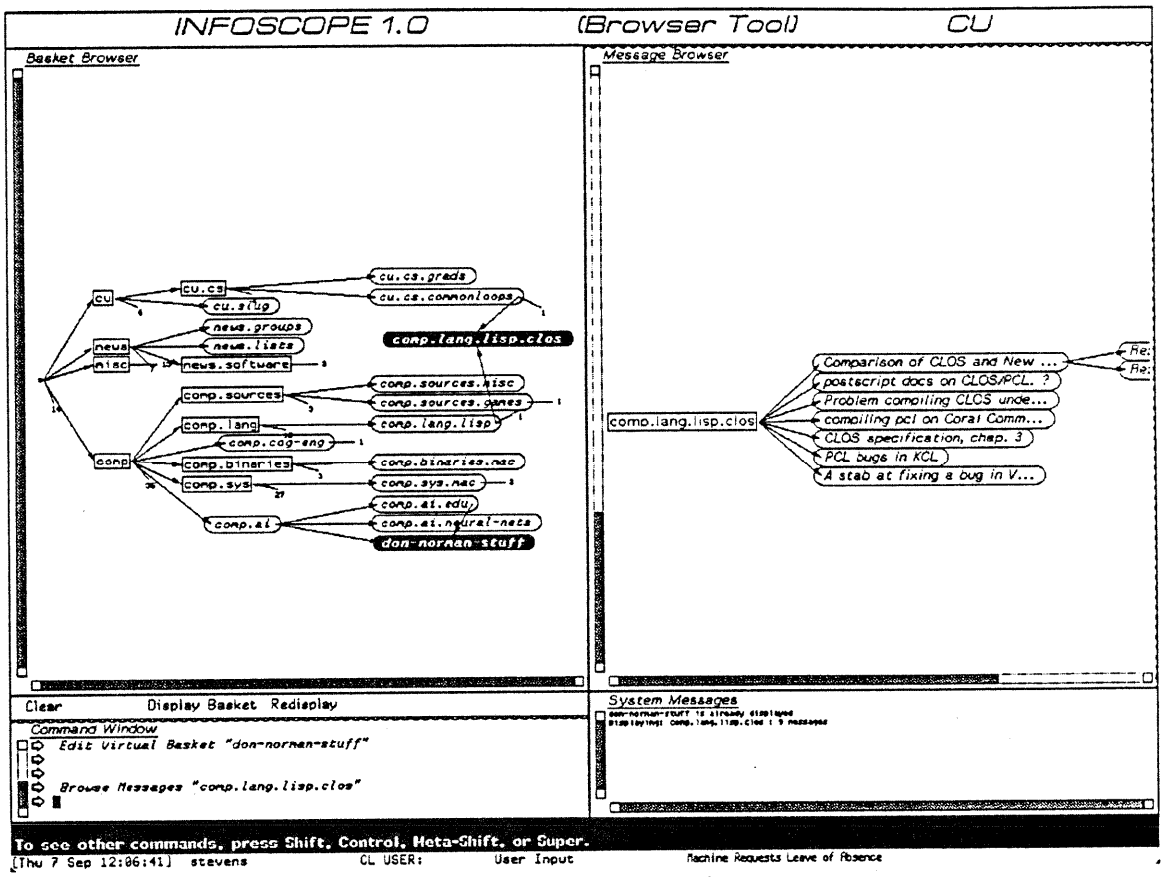
This decision to keep new data files compatible with old ones means that additional knowledge structures must be built. Perhaps the most important of these is the list of virtual baskets designed by a user. This structure contains the names of each virtual basket, the filter which defines which messages are inherited into that basket, and the set of locations (real and virtual) from which those originate. This in turn is used by the display system to build a visual representation of this structure on the screen.

Another source of useful information to the system is the displayed hierarchy itself. When users display a basket on the screen they are more than likely indicating an interest in messages from that basket. Prolonged use of that basket clearly indicates this. Therefore, the user's actions with respect organizing the display are gathered by interested agents and stored internally. Of course the virtual hierarchy itself is another valuable source of knowledge about the user. The vocabulary used to name these structures may be associated by agents with any messages filtered into that basket. The combination of the hierarchy and the virtual hierarchy constitute the main portion of the system model which the user must deal with.

Finally, there is the knowledge which is garnered by INFOSCOPE from the message storage process. Each time a message is stored it must be place somewhere in particular. This location has a path and a name which will utilize terms from individual situation models. These terms should therefore be used whenever appropriate in agent suggestions etc.

4.2 The Infoscope User Interface

The INFOSCOPE user interface is a graphical one which utilizes a modified version of the TRISTAN general graph display tool. This is the same tool which is used for



4.1 Browser Tool

The Browser Tool allows users to peruse and learn the existing structure of baskets in INFOSCOPE. Messages within a basket are also browsed in this tool. From here the user can click the mouse on objects to invoke reading, posting, or virtual structure operations.

displaying hierarchies in the HELGON and CODEFINDER systems. This means that users who are familiar with any of these system will have the knowledge necessary to browse the displayed structures. The only difference between the original graph system and the INFOSCOPE version is that additional node representations have been added to INFOSCOPE so that a clearer understanding of the semantics of a node is imparted to users.

Nodes – As shown in the basket browser (part of the browser tool, Figure 4.1) there are three different node representations in the INFOSCOPE display. The first of these are the square nodes. These are intermediate nodes in the hierarchy and are associated only with basic actions such as "move this node," "display some children," "display all children," and "redisplay the subhierarchy of this basket." Oval nodes represent baskets which have only messages in them (newsgroups). These nodes are associated with operations such as "browse messages," and "create virtual basket." The highlighted oval nodes are user defined virtual baskets and represent extensions to the USENET distribution structure. For example, in Figure 4.1 the *comp* and *comp.lang* nodes only serve to illustrate the structure of the information space. The *comp.lang.lisp* node, however, is a USENET newsgroup and therefore is an oval basket. Finally, the *comp.lang.lisp.clos* node is highlighted since it is a virtual basket consisting of selected messages (those about the language CLOS) filtered from the baskets *comp.lang.lisp* and *cu.cs.commonloops*. Virtual baskets should become the users' center of attention as they represent the areas of keenest interest to the the individual user. The relationships in the predefined hierarchy with the extended virtual structure are represented by the arrows between nodes, and the structure is no longer a strict hierarchy, but a directed graph. This can be compared to a typical text based interface to the same information space (see Figure 2.2).

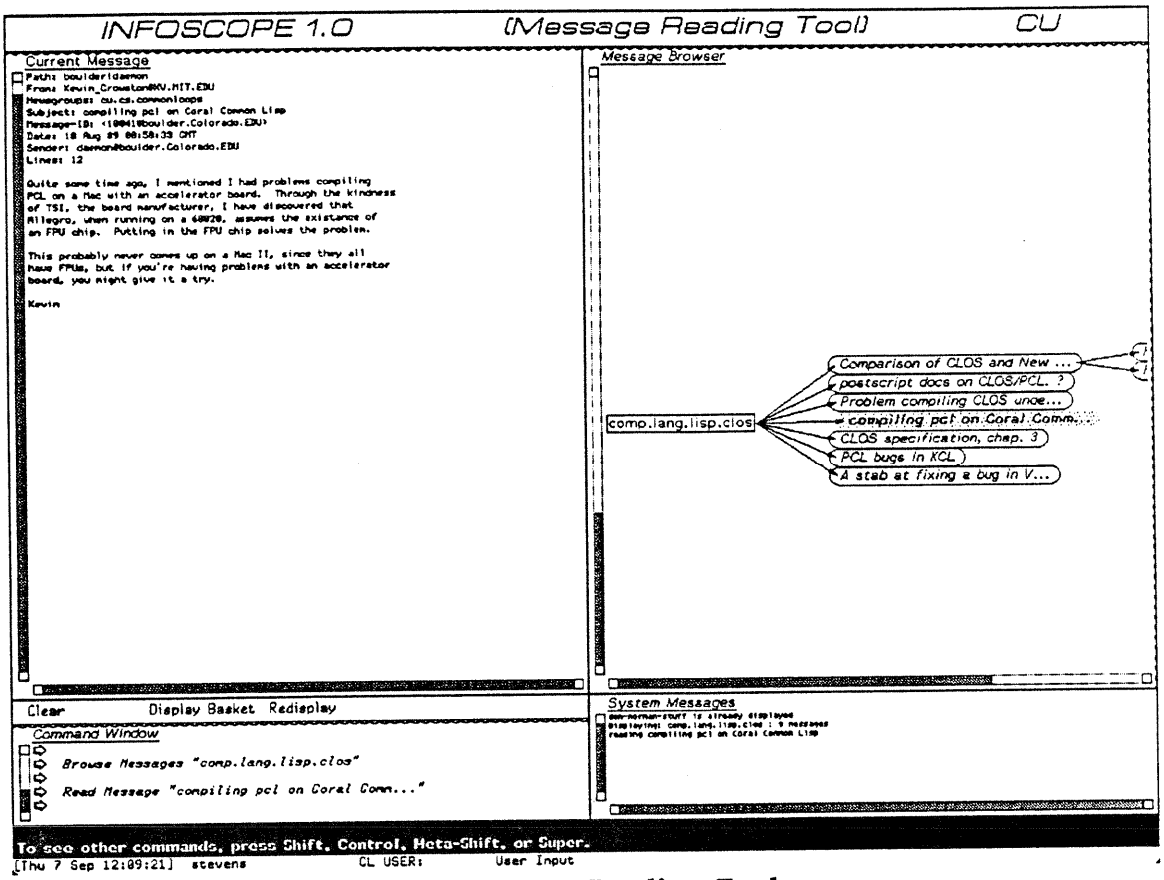
Messages – In addition to browsing and manipulating the structure of baskets in INFOSCOPE, users can use the message browser pane of the browser tool to display and navigate through the set of messages contained within a basket. The message browser pane is located on the right side of Figure 4.1 and in this example it shows the set of messages that were contained in the virtual basket *comp.lang.lisp.clos*. Notice that the node representing the virtual basket in the message browser is square, as opposed to its highlighted oval appearance in the basket browser. This is because the center of attention in the message browser is finding relevant messages for reading. When selected, objects in the message browser are displayed. However, baskets can't be displayed from the message browser since any basket in the message browser is already displayed there! A displayed basket in the message browser only serves to accent the structure of messages and therefore is treated as an intermediate node.

Another aid to the user in understanding the structure of messages within a group is the hierarchical display of those messages. In this way messages are organized around the concept of a *conversation*. A conversation consists of a message node, all responses to the message in that node, and similarly all responses to the responses until leaf nodes are reached. This idea is demonstrated in Figure 4.1 by the conversation about "Comparison of CLOS and New Flavors." Notice the two nodes which are only partially displayed, start with "Re:", and are attached to the "Comparison..." node. These messages are responses to the original posting. "Re:" stands for 'regarding' and is prepended to the subject line of a message by the standard news software whenever a reply is posted. Messages in response to a response will have "Re: Re:" at the beginning of the subject line and the process continues like that as long as the conversation flourishes.

There are several advantages to displaying the structure of messages in this manner. In the INFOSCOPE system conversations may be treated as an object. This means that operations such as "Save this conversation," or "Monitor this conversation" can be used to write all of the messages of a conversation to a file at once. If a conversation is monitored then new messages in the conversation are automatically saved to the same file as the original conversation when they arrive. Another advantage is that this makes saving of subconversations as simple as saving entire conversations. For example, if someone responds to a posting with a particularly interesting comment it may start a whole new conversation within a conversation (a new thread). This thread can be saved as a conversation by clicking the mouse on the node at the root of the thread (ie., the message with the interesting comment). Also, since some interesting threads within a conversation will result in deep message hierarchies starting from certain nodes (ie., lots of responses), the display of this structure can immediately lead users to these interesting threads.

4.3 A Scenario Using the System

The INFOSCOPE system is arranged around the four tools mentioned above. The first tool which users will typically encounter will be the browser tool (see Figure 4.1). With this tool users can explore the structure of baskets in the information space. Initially, the system will display a combination of some general predefined hierarchy and the baskets which are commonly perused by the individual users. When users choose a basket to browse they click the mouse on that node and select the action "browse messages" from the resulting menu. Doing so results in the display of the messages within that basket. For example, in Figure 4.1 a user has asked the system to display the messages from the comp.lang.lisp.clos virtual basket. These messages are organized by the conversational metaphor already discussed.

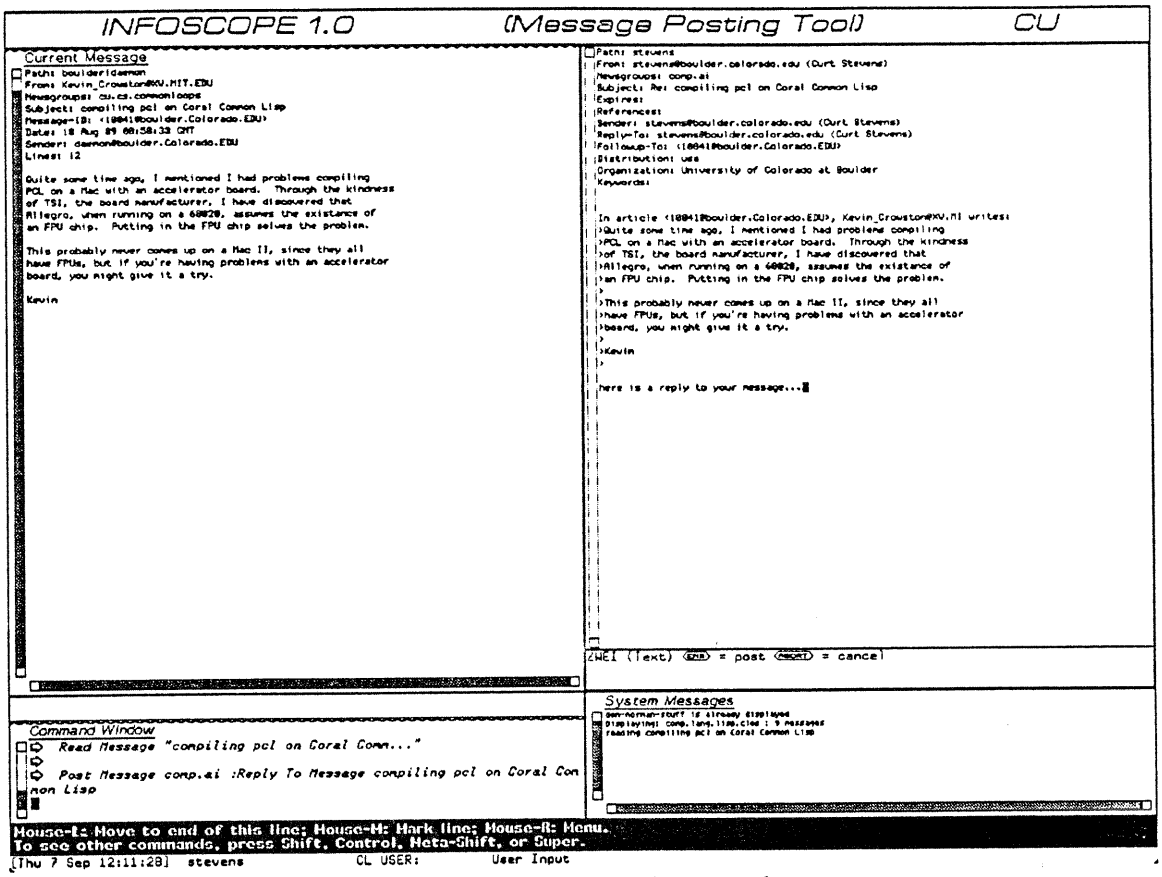


4.2 Message Reading Tool

The Message Reading Tool allows users to select messages for reading as well as other standard operations on messages like saving and deleting.

Once an interesting message has been identified by browsing the subject lines in the message browser, the next step commonly taken is for users to read that message. This is accomplished by clicking on the appropriate message node and selecting "read message" from the resulting menu. When this occurs INFOSCOPE automatically switches users into the Message Reading Tool (see Figure 4.2). In order to keep the context of interaction with users, the message reading tool also contains the same message browser window that is in the browser tool. This also makes it easier for users to select the next message they wish to read. Alternately, users can type a control character and INFOSCOPE will display the next message from the current conversation (or the first message from the next conversation if the current conversation is already completely displayed). Notice that there is one node "dimmed" in the message browser. This represents a message which has already been read. The current message window not only contains the current message, but the entire history of read messages in a scrollable list above the current message.

After browsing and reading several messages users may wish to post a response to a particular message. This is accomplished by clicking the mouse on either the messages node in the message browser or on text of the message itself. Selecting "Post a Followup to..." from the resulting menu brings users into the message posting tool (see Figure 4.3).

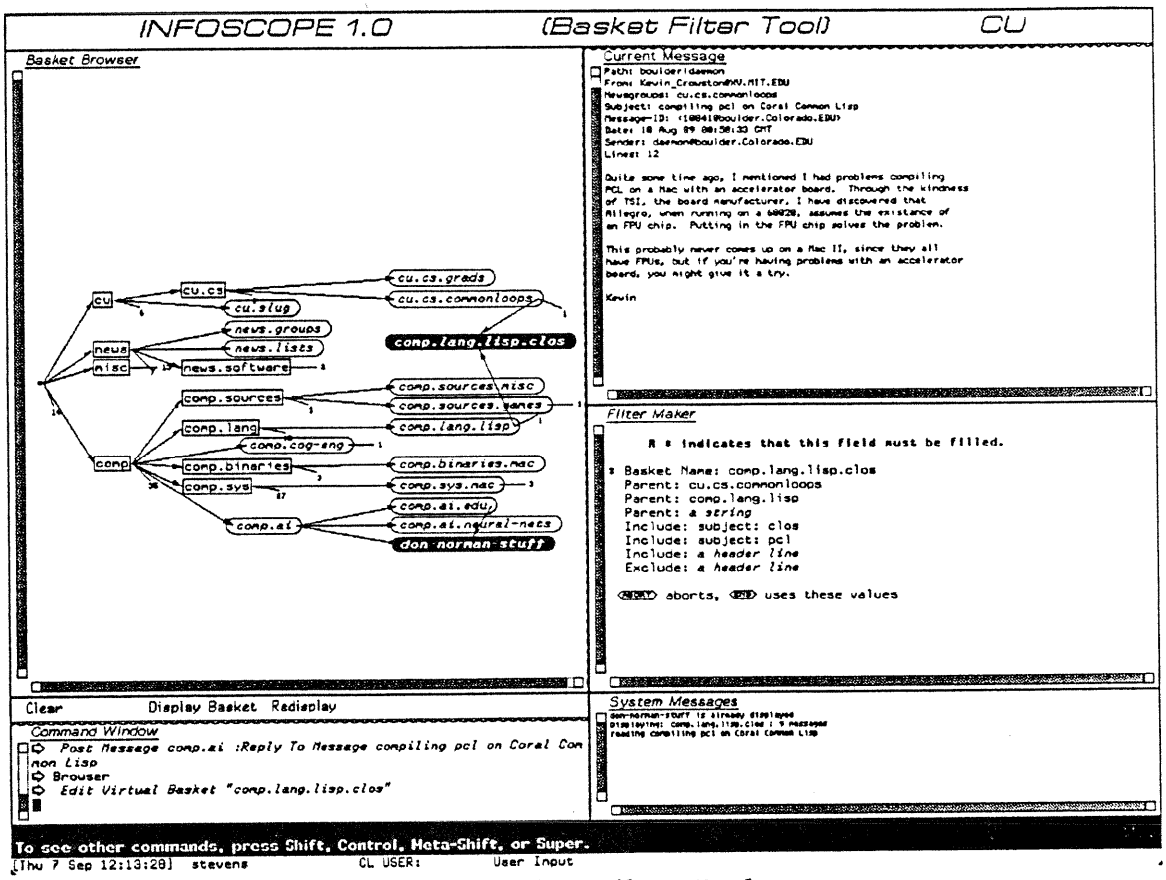


4.3 Message Posting Tool

The Message Posting Tool appears when users ask INFOSCOPE to post original or follow-up messages to a basket. Of course INFOSCOPE figures out the appropriate place for messages in virtual baskets to be posted. On the left is the message browser so that users can refer to previously read messages when composing the posting. The message being replied to is available in the kill buffer and is place in the editing pane on the right.

This tool also helps maintain context by retaining the current message window from the message reading tool. The other window (the one on the right–hand side) is a fully functional editor which allows user to compose and modify their postings. When users wish to reference the message that prompted their followup posting they can easily include that message into the editor. When the followup message is completed users simply press the **END** key and INFOSCOPE posts the message to the appropriate newsgroup. This may require some calculation since the followup may originate in a virtual basket. When this is the case INFOSCOPE determines which real newsgroup the original message was posted to and then constructs the followup message so that it returns to this group. The fact that users are not posting to virtual baskets is completely transparent to them.

In order to browse or post to virtual baskets they must first be created. This process takes place in the Basket Filter Tool (see Figure 4.4). This tool consists of the current message and basket browser windows described already, plus one additional window, the filter maker. In order to initiate the process of creating virtual structure users click the mouse on any basket which will be one of the parents of the virtual basket. Parents can be either real or virtual baskets. When users select either "Create Virtual Basket" or "Edit Virtual Basket" they are taken into the filter tool and presented with a partially filled out filter. INFOSCOPE uses whatever information it has to fill in the filter. In the case of creating a new basket, it fills in a reasonable name for the new basket as well as the parent which was used to enter the restructuring process. In the example of Figure 4.4 users are editing the definition for the virtual basket comp.lang.lisp.clos which already existed. This particular basket is defined to



4.4 Basket Filter Tool

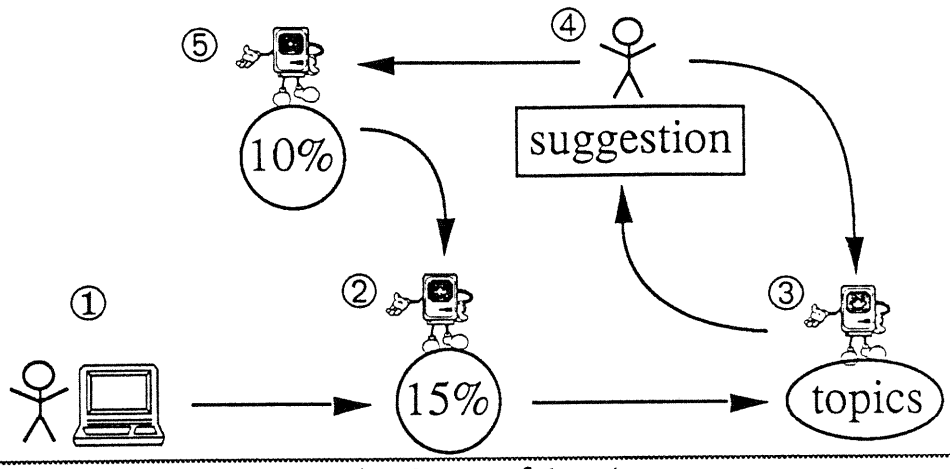
The Basket Filter Tool allows users to create and modify the filter descriptions for virtual baskets. In this example, the virtual basket *comp.lang.lisp.clos* is being edited. You can see that this basket inherits messages that contain the subjects **pcl** or **clos** from the existing baskets *comp.lang.lisp* and *cu.cs.commonloops*.

inherit messages from the baskets comp.lang.lisp and cu.cs.commonloops, and specifically to include any messages containing either of the strings "clos" or "pcl" in the subject line of the header field. Using this new virtual basket users now have a repository for information about CLOS which is displayed using the name semantically attached to collections of that type of information by the users who defined the basket. Each individual user has their own set of these virtual baskets defined.

4.4 Agents

The agents in INFOSCOPE are a collection of rule-based heuristics which utilize user models to facilitate the creation of suggestions to the user. These suggestions will concern the future modification of the system model which the agents deem is necessary for proper evolution of the structure. Users can then accept, reject or modify these suggestions and evolution proceeds in this cooperative fashion (see Figure 4.5). In addition, the agents analyze suggestions which are rejected or modified so that the user models can be appropriately modified. When a suggestion is rejected, something should be done to ensure that the same error in judgement is not included in future suggestions which occur due to similar circumstances. The agents which do this analysis of suggestions and modification of other agents are called supervisory agents. They do not modify *what* the agents do, but why and when they do it.

It has been suggested that the introduction of agents into computer systems can be harmful [Wroblewski, McCandless, Hill 1989]. This is because unlike human agents, computer agents do not have a shared understanding which can be utilized to model



4.5 A Group of Agents

At ①, the user interacts with INFOSCOPE for a period of time which extends until agent ② recognizes that the user consistently reads less than 15% of the available messages in a particular newsgroup. At this time agent ② tells agent ③ to begin gathering topics that are judged interesting and uninteresting by the user. When a sufficient volume of data has been gathered agent ③ posts a suggestion to the user. (④) If the user accepts the suggestion agent ③ continues gathering topics for modifying filters in the future. If the user rejects the suggestion, agent ⑤ tries to find out why and uses this information to modify agent ②. In this example the user complains that the agent is too intrusive. It is modified so that it only reacts when the user reads 10% or less of the messages in a newsgroup.

the knowledge state and goals of their human partners. This can lead to a breakdown in communication that leaves the human feeling stereotyped. INFOSCOPE agents avoid this pitfall by never attempting to converse with the user. Rather than implementing fully autonomous agents, we designed agents which serve as semi-autonomous helpers. From the user's point of view INFOSCOPE agents serve as a feedback mechanism, helping the user to modify the displayed structure in response to observations of their own system usage patterns. Agents post suggestions to the user which must be accepted for them to be implemented. In this model, shared knowledge about the information space is more important that a shared understanding of goals.

What kind of agents are there in InfoScope? – There are several agents which divide the task of helping the user into logical sub-tasks. For example, there are agents watching for conditions which dictate the creation of new baskets, an agent watching for new user preferences etc. In either case the user will perceive only a single InfoScope agent which in either an active or passive fashion will assist the user in carrying out their information tasks.

What knowledge does an agent have? – An agent is really a set of heuristics and/or rules which define knowledge about how to use the information, semantics and statistics which are kept in InfoScope objects (where objects are groups, messages, the user model, etc.). In that sense, the data contained in InfoScope objects is also part of whatever agent utilizes it. In this way data is shared among many different agents without being replicated (really the standard object oriented approach).

The agent which creates new baskets, for example, gets its information from various sources. Statistics which are kept in the objects representing each basket (newsgroup, mail folder...) supply the basket-agent with details of how often the basket is perused, what kind of messages are read and ignored (ie., simple semantics like keywords and keyword pairs), and what kind of messages are inherited from this basket and displayed in other baskets. The agent also gets certain information from the user model. Some users may prefer many baskets with highly refined contents while others might prefer coarse filters on baskets which lead to diverse groups of

messages. Still other users may choose not to utilize the basket creation facilities at all, and this information is located in the user model. The user model, in turn, may be modified by an agent consisting of heuristics and rules which watch for shifting in user preference patterns. Observations made by this type of agent would usually result in suggestions or questions to the user which would evolve the user model.

Agents have characteristics as well as rules and available information. In particular, agents have *talents* and a *personality*. An agent's talents are defined by the rules and heuristics which make up the agent. The groups of rules allow the agent to carry out specific tasks and analyses and these comprise the agent's talents. Personality defines how the agent interacts with the user of the system. One important aspect of personality is whether the agent is active or passive, and if active, how intrusive the agent is. Other components of personality relate to the user model contained in the system for each user. If most of the agent's suggestions to the user are ignored or aborted then the agent should change its personality offer less help to the user. The idea is not to give users the feeling that agents supply a lot of useless suggestions.

Who creates agents? – Currently the intention is to have all agents created a priori as the system is developed. While there are no plans to allow users to create their own agents, the agents which are supplied with the system are user modifiable and in fact are affected by *supervisory agents* (agents which modify the personality of other agents based on observed changes in the user model etc.). This does not exclude the possibility of adding user created agents in the future, but more analysis would be necessary to determine what that would entail.

Do agents change over time? – An agent's personality changes over time as described above. Agent personalities are available to the user for editing at any time. Its talents, however, do not change over time. Since adding talents requires the addition of a robust set of heuristics/rules, allowing the user to do so at will is a difficult prospect [Fischer, Girgensohn 1990].

What metrics do agents use to determine *interesting* and *uninteresting* messages? – The main metric used for this task is matching keywords and keyword pairs. These keywords are gathered in two major ways. Many of the keywords which determine (un)interesting messages are supplied directly by the user through access to the user model (the InfoScope concept counts on a reasonably cooperative attitude on the part of users). Other keywords are gathered by an agent which scans messages for semantically important words. Words from read messages contribute to a list of interest indicators while words from unread messages contribute to a list of disinterest indicators. Conflicts between these lists lead to requests for assistance from the user in the form of an active query of passive suggestion/question on the suggestion agenda. Some use of latent semantics analysis may be included in this process as its potential role becomes clearer.

4.5 User Modelling

The user models in INFOSCOPE are concerned with two main aspects of system usage. The first of these will deal with the user's individual preferences with respect to the daily operation of the system. This includes which baskets should be initially displayed by default, whether entire subhierarchies of conversations should be displayed by default or only on demand, where the system should read the newsrc from, how often the news stream should be reset and baskets updated to their new state and many other aspects of day to day operation.

The second aspect of system usage which will be monitored by the user model is the actual content of information which is deemed *interesting* or *uninteresting*. This task is basically a statistical one which requires the model to keep track over long periods of time of what users are reading and what they are ignoring. This happens on two levels. At the higher level the system monitors the amount of interesting information which is found on a regular basis within a specific basket. When the amount is too low, steps should be taken to modify the existing structure. At the

lower level the topics, or concepts, covered by a message are retained in order to facilitate the evolutionary process in a semi-automatic fashion.

From these two parts of the user model, INFOSCOPE can learn much relevant information about individuals who use it on a regular basis.

5. Limitations, Evaluation, and Future Work

INFOSCOPE is about to be installed for regular use by the members of our research team as well as distributed to an external site. From this use we will be gathering extensive data about filter definitions and the resulting structures for some time. This information will in turn be evaluated so that we may recognize more of the limitations and areas for future investigation. However, there are limitations which are apparent even prior to this evaluation.

One drawback of the INFOSCOPE system is that it doesn't have the ability to comprehend the content of messages. Current the system works entirely in the header fields of a message. Since vocabulary in the subject line may not match the vocabulary of the message body, some relevant messages are missed by certain filters.

Also, INFOSCOPE'S ability to define filters is limited to a small subset of boolean logic. This means that some expressions are difficult to make. It is not even clear that boolean login is the best way of defining these filters. For example, a technique called Latent Semantic Indexing has proven better than keyword techniques for clustering messages in the news domain [Foltz 1990]. Enhancing the filtering capabilities is one area of necessary future work.

We are interested in exploring how successful we can be in using a knowledge poor system. INFOSCOPE is knowledge poor since it does not understand message content or user goals. Once we know more about this issue we will be adding certain types of abstracted knowledge into the agents. For example, INFOSCOPE will eventually have an understanding of the concepts of time, organizations (like HCIC), and persons (like in the HELGON system).

Another question concerns how much structure users of INFOSCOPE will actually be willing to supply. It is hoped that the introduction of agents will significantly increase this amount.

Finally, we will investigate the issues involved with integrating the INFOSCOPE system with the CODEFINDER and EXPLAINER systems. Together these systems represent a comprehensive tool for accessing the highly complex information spaces we have been discussing.

References

[Carroll, Rosson 1987]
J.M. Carroll and M.B. Rosson, *Paradox of the Active User*, in J.M. Carroll (eds.), *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*, The MIT Press, 1987, 5.

[Dijk, Kintsch 1983]
T.A.v. Dijk and W. Kintsch, *Strategies of Discourse Comprehension*, Academic Press, New York, 1983.

[Fischer, Girgensohn 1990]
G. Fischer and A. Girgensohn, *End-User Modifiability in Design Environments*, Human Factors in Computing Systems, CHI'90 Conference Proceedings, ACM, Seattle, WA, April 1990.

[Fischer, Henninger, Redmiles 1990]
G. Fischer, S. Henninger and D. Redmiles, *A Conceptual Framework and Innovative Systems for Accessing Knowledge for Software Reuse*, Proceedings of the HCIC Consortium 1990 Winter Conference, HCIC, Sna Diego, CA, February 1990.

[Fischer, Nieper-Lemke 1989]
G. Fischer and H. Nieper-Lemke, *HELGON: Extending the Retrieval by Reformulation Paradigm*, Human Factors in Computing Systems, CHI'89 Conference Proceedings (Austin, TX), ACM, New York, May 1989, pp. 357-362.

[Foltz 1990]
P.W. Foltz, *Using Latent Semantic Indexing for Information Filtering*, Proceedings of the Conference on Office Information Systems, ACM, Cambridge, MA, 1990.

[Grudin 1989]
J. Grudin, *Why CSCW Applications Fail: Problems in the Design and Evaluation of Organizational Interfaces*, Office: Technology and People, Vol. 4, No. 3, 1989, pp. 245–264.

[Hutchins, Hollan, Norman 1986]
E.L. Hutchins, J.D. Hollan and D.A. Norman, *Direct Manipulation Interfaces*, in S.W.Draper. D.A. Norman (eds.), *User Centered System Design, New Perspectives on Human-Computer Interaction*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1986, Chapter 5.

[Malone, Grant, Turbak 1986]
T.W. Malone, K.R. Grant and F.A. Turbak, *The Information Lens: An Intelligent System for Information Sharing in Organizations*, Human Factors in Computing Systems, CHI'86 Conference Proceedings (Boston, MA), ACM, New York, April 1986, pp. 1-8.

[Moran 1983]
T.P. Moran, *Getting into a System: External-Internal Task Mapping Analysis*, Human Factors in Computing Systems, CHI'83 Conference Proceedings (Boston, MA), New York, December 1983, pp. 45-49.

[Nieper 1985]
H. Nieper, *TRISTAN: A Generic Display and Editing System for Hierarchical Structures*. Department of Computer Science, University of Colorado, 1985.

[Norman 1986]
D.A. Norman, *Cognitive Engineering*, in S.W. Draper. D.A. Norman (eds.), *User Centered System Design, New Perspectives on Human-Computer Interaction*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1986, Chapter 3.

[Reddy 1983]
R. Reddy, *Technologies for Learning*, Computers in Education: Realizing the Potential, Report of a Research Conference, Washington, D.C., August 1983, pp. 49-60.

[Schwartz 1989]
M.F. Schwartz, *The Networked Resource Discovery Project*, Proceedings of the IFIP XI World Congress, San Francisco, California, August 1989, pp. 827-832.

[Simon 1983]
H. Simon, *The Computer Age*, Computers in Education: Realizing the Potential, Report of a Research Conference, Washington, D.C., August 1983, pp. 49-60.

[Wroblewski, McCandless, Hill 1989]
D. Wroblewski, T. McCandless and W. Hill, Agency Considered Harmful, Knowledge-based Human Computer Communication Symposium Position Paper, 1989