

# Semantics Happen: Knowledge Building in Spatial Hypertext

Frank Shipman, J. Michael Moore, Preetam Maloor, Haowei Hsieh, Raghu Akkapeddi  
Department of Computer Science and Center for the Study of Digital Libraries  
Texas A&M University  
College Station, TX 77843-3112 USA  
+1 979 862 3216  
shipman@cs.tamu.edu

## ABSTRACT

Hypertext represents ideas through chunks of text or other media interconnected by relations, typically navigational links. The similarity to knowledge representations such as frames and semantic nets has led to much effort in using hypertext systems for knowledge representation and extending hypertext systems to make them able to express more. This work has met with limited success due to difficulties including the tacit and situated nature of much knowledge. Instead of viewing knowledge expression as an all at once event, we view it as a constructive process, i.e. knowledge building. The Visual Knowledge Builder (VKB) lets users express content via visual or textual means and later formalize that content in the form of attributes, values, types, and relations. VKB proactively supports this process through a set of suggestion agents whose interaction with the user is mediated by the suggestion manager. Preliminary evaluation of the suggestion manager and suggestion agents yields positive results but further confirms that there is no “silver bullet” for knowledge engineering -- semantic expression is most likely to happen during, and is driven by, task performance.

## Categories and Subject Descriptors

H.5.4 [Information Interfaces and Presentation]: Hypertext/Hypermedia - architectures, user issues

## General Terms

Design, Human Factors

## Keywords

spatial hypertext, suggestion-based interfaces, mixed-initiative dialogs, visual language, spatial parser, incremental formalization

## 1. HYPERTEXT AND KNOWLEDGE REPRESENTATION

Hypertext has long been recognized as a form of knowledge representation emphasizing interdocument relations. In addition to more common page-sized nodes found in systems like KMS [1] and the Web, a number of systems have included smaller-sized information chunks to represent finer-grained relations. Early examples include the representation of Searle’s Chinese Room argument in Euclid [37], Toulmin-structured analyses in

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

HT’02, June 11-15, 2002, College Park, Maryland, USA.  
Copyright 2002 ACM 1-58113-477-0/02/0006...\$5.00.

NoteCards [18], and IBIS-based design rationale in Mikropolis [22] and gIBIS [4].

This first round of representational hypertext helped motivate the design of a number of systems with more expressive representations. IDE [15] extended NoteCards to include a *template card* for defining types through the inheritance of attributes and behavior within the context of instructional design. SPRINT [3], with an aim to enhancing business practice, used SmallTalk methods attached to a frame-based representation to integrate hypermedia, semantic network, and expert system representations. Concorde [11] supported knowledge engineering through a representation in which relations between nodes could be constrained. Knowledge structuring was the goal of Aquanet [19], which allowed users to define representational schemas that included object and relation types, attributes, and constraints. The above representations emphasize the expression of declarative knowledge. Phidias [23] embedded procedural knowledge in the hypertext representation, such as inferences based on forward-chaining, by allowing nodes in its hyperbase to contain a stored query. All of these systems blended representations from hypertext and artificial intelligence. Further discussions of the design and use of such systems are found in [16], [31], and [25].

Work on knowledge-based hypertext became less common with the emergence of the Web. One reason is that the Web’s primary representations, e.g. HTML, did not include useful facilities for more formal representation. More recent standards have changed that. Indeed, many of the above themes are being revisited in discussions of the “Semantic Web.” XML includes many of the characteristics of earlier efforts with regards to integrating frame-based representations and hypertext. Like Aquanet’s representation, it is a meta-language that can be used to encode specific knowledge representation languages. The Resource Description Framework (RDF) and DAML+OIL follow the knowledge interchange format (KIF) and knowledge query and manipulation language (KQML) as standards for sharing common procedural representations of knowledge.

These are all representations designed to aid in the use and sharing of knowledge once it is represented, but how the knowledge is “authored” in the first place is not considered. Our focus is on how to support this expression, which we call *knowledge building*. This is a constructive activity where the author’s own knowledge is impacted by the expression process. Besides providing visual, spatial and textual means of communication, we are building proactive support for this process.

The next section discusses difficulties with knowledge representation and implications for developing tools to help

knowledge expression. This is followed by a description of our approach of knowledge building through incremental formalization. This approach is instantiated in the Visual Knowledge Builder (VKB) through its suggestion agents and suggestion manager. Both of these components are described in detail with initial evaluation of the individual components. The paper concludes with a discussion of outcomes, open issues, and areas for future work.

## 2. KNOWLEDGE BUILDING

Expression of knowledge in formal representations has been the role of professional knowledge engineers working with domain experts on a particular system with specific goals [5]. The difficulty of sharing the resulting knowledge can be seen in the results of the CYC project, where merging knowledge bases required considerable effort for questionable value [12]. These difficulties are often social, e.g. the vocabulary problem [10], and exist regardless of the underlying representation. The end result is that knowledge engineering is an expensive process that generates content specialized for a particular application [38]. This implies limits on both the situations in which knowledge engineering is economically feasible and the usefulness of sharing the results.

How might the use of a hypertext-based knowledge representation change this? One answer can be seen in Peper and colleagues replacement of an expert system for technical support with a hypertext system [26]. The advantages of the hypertext system included that it was easier to modify than the knowledge base and that people could apply their knowledge of the situation and common sense in deciding how to interpret the contents. The power of a semi-formal representation like hypertext is that it can allow the system to do what it does best (store, filter, and retrieve) while the human does what they do best (determine how and when to act on available information) [17]. The weakness is that, when there is too much information for a person to consider, the system is limited in the types of support it can provide.

Thus, integrating hypertext and more formal knowledge representations has the potential for combining ease of expression and manipulation of content in less formal representations with greater system support based on formalized content.

But will people express content in a formal representation in such systems? Not initially. Reasons include the additional cognitive overhead required to learn the representation, the fact that much of people's knowledge is tacit [27], and finally that the abstraction of knowledge into a formal representation requires anticipation of how that knowledge will be used [38]. For a more thorough discussion of the difficulties of formal expression, see [34].

For these reasons, knowledge expression (or from the system's perspective "knowledge acquisition") is really knowledge building. Knowledge expression implies that the knowledge exists prior to its externalization. The effort to learn the formal representation, to uncover tacit knowledge, and to anticipate the contextualized use of that knowledge changes the person's understanding. This change in understanding takes time and is intertwined with the act of expression. For people to express their knowledge, systems have to support an incremental knowledge building process.

In spatial hypertext, at least in analytical and organizational uses of spatial hypertext, knowledge building takes the form of deciding on visual representations for information chunks and their interrelations. This results in the definition, use, and

evolution of a visual language. But the language is not prescriptive or restrictive, it provides an advantage to spatial hypertext where tacit knowledge and unanticipated uses are common.

## 3. PROACTIVE SUPPORT FOR KNOWLEDGE BUILDING

Our general approach to knowledge building is one of incremental formalization. This process proceeds through three phases:

1. Users collect, author, manipulate, and revise visual information spaces containing semi-structured information.
2. As they manipulate the information, the system uses a variety of AI techniques to recognize structures that may be semantically meaningful.
3. The system provides access to these structures and makes suggestions based on these structures.

Suggestions for formalization provide assistance in moving the user constructed informal representations of data into more formal representations that a computer can process.

Users can benefit from formalization suggestions in different ways. Background processing can help analyze large information spaces that might be boring or time consuming for users to process themselves. Identified potential formalizations provide a savings in time, effort and cognitive load for users. Additionally, the heuristic processing might identify patterns that would otherwise be missed by users. Finally, they can provide short cuts to accomplishing a formalization that users already know about, but have not yet done. The suggestion provides a way to easily effect the change, potentially saving time.

### 3.1 Suggestions

A suggestion is something offered for consideration. The idea of a suggestion implies that the suggestor does not know that the suggestion must happen and consequently leaves the decision to the recipient. When a person or a computer makes a suggestion, persons receiving the suggestion can choose to accept or reject the suggestion. They may also choose to take no action at all. We identify two classes of help that a suggestion can provide: task assistance and tool assistance.

#### 3.1.1 Task assistance

Suggestions that provide task assistance can identify laborious tasks and offer to assist in the completion of those tasks. They can also perform automated processing of information providing options that reduce the cognitive load placed on users. Examples of this type of suggestion include when the MS Office Assistant suggests that you are writing a letter in Word and offers to help or, in PowerPoint, suggests that you should capitalize all bullets. Examples within the context of knowledge building include the conversion of a recognized structure into a formal representation. This has been the focus of most of our effort.

#### 3.1.2 Tool assistance

Suggestions providing tool assistance educate the user about processes and features available in the tool that may make work easier to accomplish. The growth of what Fischer calls "high-functionality systems" implies users can no longer be expected to know all of the features available that may be useful for their current task [6]. By suggesting faster methods for invoking commands and identifying alternative commands, users may learn

about the application while doing real work. This need not be limited to introduction and explanation of individual features, but also helps in combining features to become an “expert user.” Most proactive help systems would fall into this category. An example of this type of suggestion is when Activist, an active help system for an EMACS-like editor, analyzes user work to identify suboptimal behavior and suggests alternatives [7]. For knowledge building, tool assistance includes help with the formal representation and interfaces surrounding it.

## 3.2 Issues with Suggestions

Using suggestions as a mechanism for assisting users requires examination of issues such as work efficiency, notification, distraction, trust and user control. These issues are related to and overlap those of agents [39] and mixed-initiative dialogs [13].

### 3.2.1 Work Efficiency

An assumption for using any tool is that there is some kind of efficiency of work that is achieved that could not happen otherwise. Discussion about each class of suggestion has already pointed to benefits that affect efficiency such as reduction of cognitive load and time savings. The way the notification mechanism interacts with users can affect work efficiency as well.

### 3.2.2 Notification

Discussion of the notification system must begin with a discussion of whether the generation of suggestions will be proactive or reactive. While we are primarily concerned with proactive systems, the inclusion of suggestions does not necessitate that a system be proactive. If notification is reactive, suggestions are generated only when specifically requested by users. Proactive notification occurs when the computer generates suggestions and informs users during their work. An intermediate option utilizes a mixed-initiative dialogue. A mixed-initiative is where either the computer or user can initiate suggestion presentation [13]. This allows for automated background processing to result in appropriate user notification while still allowing users to request suggestions as needed.

Persistence of notifications and suggestions is a related issue. Many systems expect suggestions or help to be attended to immediately or at least prior to the appearance of the next suggestion. This can impact efficiency since users may have to interrupt work in order to deal with a suggestion or miss suggestions that are replaced.

Often the notification method is all or nothing - it either contains all the details of the suggestion or only that a suggestion exists. Showing all the information during initial notification is more likely to distract the user while no information removes the ability for users to evaluate how to proceed in the context of their task. Progressive disclosure, or providing more information as users show more interest in the suggestion, can alleviate this problem. This allows users to assess whether they want to deal with the suggestion at that moment or not.

### 3.2.3 Distraction

The degree to which a suggestion will be distracting depends on both the user’s state when notified and how the user is notified. Disruption of users’ work is not an issue with reactive systems. Nevertheless, in a mixed initiative system, proactive notification can be problematic.

When users work, they can enter into a highly focused state. While in this state any distraction or interruption can interfere with the work being done - even if the interruption or distraction is

related to a potentially useful suggestion. Schön describes professional action as being unselfconscious action punctuated by breakdowns that cause reflection-in-action [30].

Some help systems pop up a window that takes the focus. This is extremely disruptive, especially if users are highly focused on a task. By taking the focus, users are forced to deal with the help. This is the “fire alarm” approach to notification and, in the context of suggestions, should be used sparingly if at all. Ideally a system should inform users without being intrusive. If the “suggestion” is exceedingly important, the application’s equivalent to “get out of the building because it is on fire”, then such intrusion may be desired [39].

Distractions do not necessarily have to come through the notification process. In an effort to convey information over time or to increase trust, animation or sound may be introduced between notifications. For example, the MS Office Assistant fidgets and blinks to appear more lifelike. However, this can prove to be distracting. The human eye is drawn to motion and thus to animation. When the animation is accompanied by meaningful information, it becomes an effective tool for giving information. When the animation does not accompany information it risks becoming a distraction or causing the user to become habituated to that motion. In this case users may not notice when the animation does carry relevant information.

### 3.2.4 Trust

When suggestions are presented, users’ trust impacts whether or not the suggestion will be accepted or even considered. The heuristic methods used to generate suggestions have the potential to provide false suggestions. These could be task or tool assistance suggestions that do not make sense in the current context. If too many false suggestions are generated, users may not want to deal with suggestions or trust the validity of any suggestion.

### 3.2.5 Control

Another important consideration when providing user assistance is control. As noted earlier, the paradigm of a suggestion already frames information in a form that puts users in control. The computer is not saying how it *should* be, but providing options of how it *could* be.

Providing users other kinds of control can alleviate many of the issues identified here. For example intrusive notification can be undesirable unless it is the method a user selected for notification [39]. Furthermore, control over the process of suggestion generation can give users the ability to tailor the options to fit their individual preferences and needs.

So far, the discussion of suggestion generation and notification has been domain independent. The next section describes prior experience with suggestions within the context of knowledge building.

## 3.3 Prior Knowledge Building Suggestions

The Hyper-Object Substrate (HOS) integrated a navigational hypertext with a frame-based knowledge representation [36]. As users authored pages with text and graphics, the system looked for textual cross-references between information chunks and made suggestions for attributes and relations based on these cross-references. Attribute and relation suggestions for an information chunk would appear when the user opened the interface showing already defined attributes and relations.

VIKI deliberately did not include relations in its formal knowledge representation but did include object types,

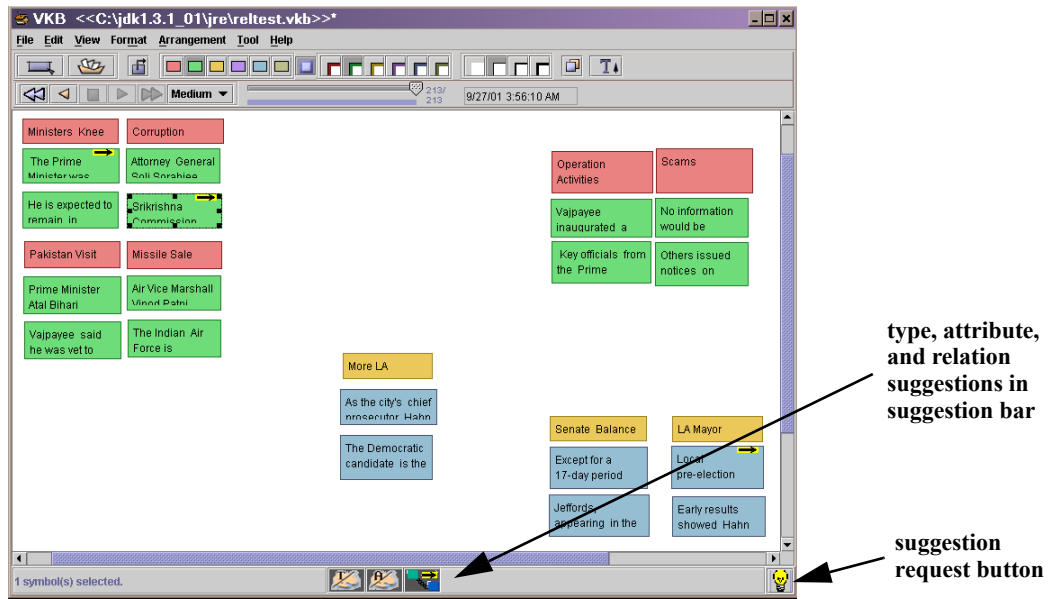


Figure 1: The VKB workspace with suggestion request icon in lower right corner and three suggestions in the suggestion bar.

collections, and composites [20]. VIKI included proactive suggestions for assigning object types when an object's attributes significantly matched those of a defined type. VIKI also included user requested suggestions for new collections and composites based on the visual structure recognized by its spatial parser [35].

#### 4. THE VISUAL KNOWLEDGE BUILDER

The Visual Knowledge Builder (VKB) is a system for gathering knowledge in a variety of forms and representations. VKB users create a hierarchy of two-dimensional spaces called collections that can contain other collections and visual symbols. Symbols are the visual representation of information objects. The placement of symbols into collections is similar to the placement of files into folders in the Windows/Macintosh operating systems.

Users modify visual attributes for collections and symbols to indicate their interpretation of the information. Expression of relationships and categories occurs by placing symbols near one another or placing symbols in a collection. Interacting with information in VKB is the visual manipulation of symbols and collections in the workspace. Manipulations include affecting position, size, border width, font style, transparency, as well as colors for the background, font, and border of symbols and collections. Figure 1 shows a VKB information workspace. A more complete description of VKB can be found in [32] and [33].

VKB's proactive support for knowledge building is composed of a set of suggestion agents and a Suggestion Manager. Figure 2 shows the communication between the user, acting through the VKB interface, suggestion agents and, the Suggestion Manager. The next section presents the VKB Suggestion Manager. Following that is a description of the suggestion agents.

#### 5. THE VKB SUGGESTION MANAGER

The suggestion manager in VKB supports multiple suggestion agents and manages these agents and the suggestions generated. The suggestion manager is designed to manage a growing number of suggestion agents.

The suggestion manager does not present itself as a help system. Rather it falls under the category of intrinsic support. Intrinsic

support is "so integrated into the interface structure, content, and behavior and the application logic that it is impossible to differentiate it from the system itself" (by Gloria Gery in [29].) The suggestion manager is an active part of the entire VKB system, so it does not feel like help. Rather, it is just another part of the system.

Unlike many help systems, suggestions in VKB are maintained in a suggestion history so the suggestions become persistent artifacts throughout the lifetime of a VKB document. This frees users from having to deal with suggestions immediately since they can always view past suggestions and address them (or not) at their convenience. As indicated earlier, the suggestion paradigm provides users control over which suggestions are implemented.

When a suggestion is initially triggered in VKB, there may not be enough information for users to determine whether it should be accepted or rejected. However, as the information workspace develops, the disposition of suggestions may become clear. Users

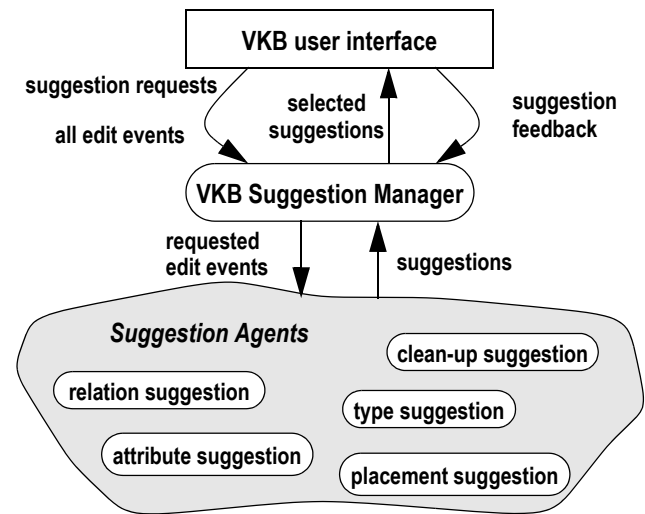


Figure 2: Suggestion manager mediates between individual suggestion agents and the user interface.

can then respond to the suggestion if it is re-presented or may use characteristics of the suggestion to locate and dispense with the suggestion via the suggestion manager.

The VKB suggestion manager uses a mixed initiative approach to triggering the generation of suggestions. As users work, the suggestion manager receives edit events from the VKB workspace and shuttles them to suggestion agents registered for particular types of edit events. Figure 2 shows that suggestion agents receive edit events through the suggestion manager and provide suggestions as a result. Alternatively, users can select an object or a group of objects in the workspace and request that all suggestion agents analyze those objects. This provides both proactive and reactive suggestion generation.

## 5.1 Notification

As suggestions are generated they are presented to users by sliding a button across a suggestion status bar in the VKB workspace (Figure 1). The motion of the suggestion across the status bar provides an indication that a suggestion has been generated. The only movement occurs when suggestions are added to the status bar and when they are removed from the status bar. The icon movement accomplishes notification with minimal distraction.

The icon present on each button mediates progressive disclosure. A unique icon represents each type of suggestion. With experience, users learn to correlate these icons with their suggestion type. A quick glance at the suggestion status bar reveals the type of new suggestions. Additionally, if the mouse lingers over a suggestion in the status bar, a tool tip reveals more information about the suggestion. Users can then determine which suggestion type might warrant immediate attention.

Suggestions do not remain on the suggestion status bar indefinitely. Eventually the status bar would overflow. As time passes suggestions are removed from the status bar. Removed suggestions can be accessed via the suggestion manager interface.

Users receive a visual indication of how much time is left before a suggestion is removed from the status bar. The background of the suggestion starts as a dark color and progressively fades until the suggestion is removed. This provides users with a visual indication of when the button will disappear. Using available information, users can decide whether to view a suggestion by clicking the status bar or waiting and opening the suggestion manager interface to find and display the suggestion. Of course users can choose to never view the details of a suggestion.

Over time suggestion agents may produce multiple instances of the same suggestion. The suggestion manager recognizes this and manages the notification process so users are not inundated with the same suggestion. As new suggestions are generated, they are compared with suggestions in the suggestion history. If there are equivalent suggestions, it checks to see the last time that an equivalent suggestion was presented on the status bar.

Each time a suggestion is presented, a minimal time is set that the suggestion manager must wait before an equivalent suggestion can be presented on the status bar. The minimal time set is increased with each subsequent presentation of an equivalent suggestion. Moreover, as equivalent suggestions are presented, the time the suggestion remains on the status bar is decreased. Eventually equivalent suggestions will no longer be presented. The basic assumption behind this implementation is that if users continuously take no action on a particular suggestion, then

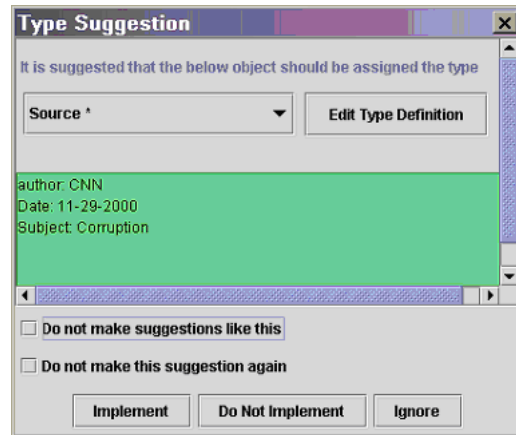


Figure 3: Suggestion interface showing a type suggestion.

eventually they do not want to see the suggestion again. However, if they ever want to find the suggestion at a later time, it is still resident in the suggestion history.

## 5.2 Suggestion Interface

When users choose to see the details of a suggestion, the suggestion is presented in a window with a standard interface (Figure 3). Each suggestion agent generates its own unique information and presentation within the top pane of the standard interface (top two-thirds of Figure 3). Users are given the option to implement a suggestion or not as well as to ignore the suggestion, thereby taking no action. Instructing the system to implement a suggestion can be viewed as delegating the task as is done with autonomous agents.

Explicitly choosing whether to implement a suggestion or not prevents equivalent suggestions from being presented in the future. Users also have the option to prevent that suggestion (i.e. equivalents) from being presented on the status bar. A final option is to discontinue presentation of all suggestions of that type. This turns off the notification for all suggestions from that agent.

## 5.3 Suggestion Manager Interface

Since all suggestions are maintained for a VKB document, the number of suggestions generated can be substantial. The history of all suggestions for a VKB workspace are presented in a table (Figure 4). The table can be sorted by column to locate suggestions with particular properties. In addition, the same options present in the standard suggestion interface can be executed in the suggestion manager when a single suggestion is selected. Users can also open any suggestion in its particular suggestion interface (Figure 3).

Through the suggestion history, users can go back and deal with suggestions when they decide it is appropriate. It also provides a way to undo previous actions such as selecting to “Never make suggestions of this Type again.” Modifying any suggestion of the type in question can undo that selection.

The second function of the suggestion manager interface is for customizing how suggestions are generated and presented (Figure 5). One option allows users to view all suggestions generated in the suggestion history or only those suggestions presented through the suggestion status bar.

Another area for customization deals with suggestion agents. Users can choose which agents are active as well as modifying

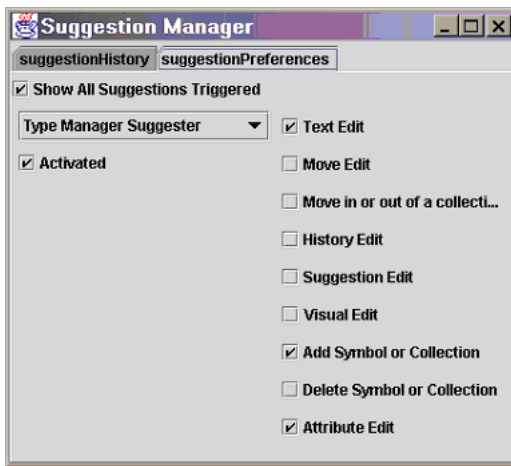


Figure 5: Suggestion manager showing configuration options.

which edit events from the VKB workspace are sent to each agent for consideration. This enables finer-grained control over suggestions than just turning them on or off.

## 6. VKB SUGGESTION AGENTS

Currently there are five suggestion agents in VKB. All provide task assistance but do so in different ways. Type, attribute, and relation suggestions help formalize knowledge representation in the workspace. Placement and clean-up suggestions provide assistance to help organize and maintain consistency within user workspaces.

- Type suggestions examine attributes present in objects and suggest assigning specific user-defined types to objects.
- Attribute suggestions identify various proper nouns present as textual context in an object. They then suggest creating attributes based on the identified proper nouns.
- Relation suggestions locate existing relations in the workspace. Using analogical reasoning, it determines whether other similar relationships exist and suggests creating those identified relationships.
- Placement suggestions examine visual structures present in the workspace to see if multiple structures concern the same semantic topic. The determination is made based on textual similarity between structures.
- Clean-up suggestions use the edit history of the workspace to determine when a structure was “broken” due to an incomplete action, such as the removal of an element from the middle of a list without closing the gap created in the list.

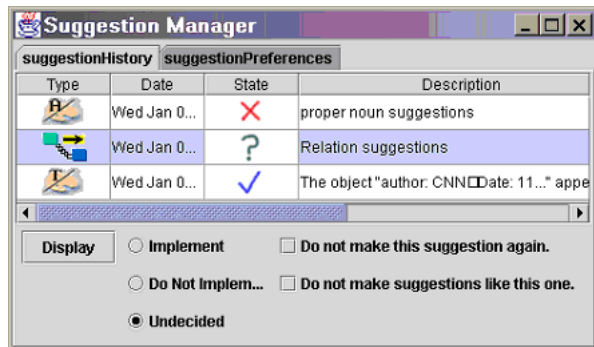


Figure 4: Suggestion manager showing history.

## 6.1 Types of Computational Analysis

Spatial, textual and temporal information about the VKB workspace is used by suggestion agents when deciding what suggestions to make.

### 6.1.1 Spatial Analysis

Much expression in spatial hypertext takes the form of relations between information chunks implied by their visual similarity and spatial proximity. To make use of this information, VKB includes a visual preprocessor to identify visually similar objects and a spatial parser to recognize structures in the layout of objects.

The spatial parser takes the layout and visual properties of the symbols in the workspace and generates a bottom-up parse of the structure. While others have subsequently used genetic algorithms [14] to recognize visual structure, the core of VKB's spatial parser uses a form of visual grammar developed by examining people's use of visual space in both computational and non-computational settings [35].

VKB extends the variety of structures recognized by the original VIKI parser by including a preprocessor to assign implicit types to symbols without an explicit type [33]. The symbols are classified based on the closeness of their visual attributes like symbol height and width, background color, border color, border width, etc.

### 6.1.2 Textual Analysis

VKB utilizes text analysis to aid the recognition of key concepts within a textual object and to determine when information objects have similar contents. The key results of this text analysis are the identification of proper nouns and measurements of textual similarity.

VKB's text analysis method is a modification of the Extractor algorithm [40]. This algorithm uses ranking strategies based on features like position, frequency, and capitalization to judge the relevance of a key phrase to the document. Document to document (and hence symbol to symbol) comparison is carried out by comparing representative term vectors.

### 6.1.3 Temporal Analysis

VKB includes an interactive history mechanism that allows readers to quickly go back and forth between the current and prior states of a workspace as well as search for a variety of authoring events on information objects. This navigable history supports (1) learning and interpreting authors' work practices, (2) recognizing patterns of activity in the information space (3) disambiguating specific actions and content.

Internally, the system stores the history events in a FIFO stack structure. The events that are logged are edit events dealing with the symbols and their visual attributes. Some of the events are Adding, Deleting, Moving and Resizing symbols, Changing background and border color and width etc. The history data structure also stores details of the events such as the symbol involved, its position and size, any color changes or changes in border width, and the state of the symbol both before and after the event is stored.

The history data enables a temporal analysis of the workspace to look for patterns in the evolution of the workspace. These patterns, created while the user is editing his workspace, may lead to old structures being destroyed and new structures being formed. The system identifies these structure changes by comparing the spatial and textual state of the workspace at different times in the workspace history.

Temporal attributes can also be used to identify recurring patterns in the user's information collection activities and use them for making predictions on what the user might do next. This concept has been dealt with in programming by demonstration literature [8][9]. While not currently part of VKB, such recognition of procedural structure can be added to the declarative structures currently recognized to further aid the suggestion process.

## 6.2 Suggestion Types

There are five classes of suggestions provided to help users formalize and maintain their information workspaces. These are: Type Suggestions, Attribute Suggestions, Placement Suggestions, Relationship Suggestions and Cleanup/Consolidation Suggestions. As the first two suggestion types are similar to prior work, they are only briefly discussed. More detailed explanations and examples are provided for the other three suggestion types. The final three suggestion types attempt to determine when a relationship exists which should be expressed, and so are spatial hypertext analogues to link finding [2] in navigational hypertext.

### 6.2.1 Type Suggestions

VKB users can define types for information objects consisting of a default set of semantic attributes and values as well as default visual properties like color and border width. When untyped objects are created, this suggestion agent checks whether they have attributes and visual properties similar to an existing type. When the object is considered similar, a suggestion to make the untyped object to be of the matched type is made. This is similar to the type suggestion algorithm in VIKI [20] but takes into account visual similarity as well as attribute overlap.

### 6.2.2 Attribute Suggestions

Much use of VKB consists of collecting text from other sources or typing in new text in visual symbols. While the system cannot understand the content of this text, it does include a part-of-speech tagger that is used to identify proper nouns. Based on heuristic categorizations of these proper nouns into categories of people, places, and things, this agent will suggest attributes for textual information objects. This is similar to the attribute and relation suggestions in the Hyper-Object Substrate [36] but is based on more advanced text analysis methods.

### 6.2.3 Placement Suggestions

VKB is designed for supporting long-term evolution of the user workspace in terms of data management and analysis. Users augment their workspace over time by adding, removing and updating information. As users add new symbols or groups of symbols into the workspace, they may not remember where similar information was previously placed, resulting in multiple visual structures about the same topic. Hence a potentially useful suggestion is to move an individual symbol or a group of symbols (e.g. list, stack) into an existing information structure. Even if not accepted, such a suggestion informs the user of similar content.

Determining when to make placement suggestions utilizes the spatial and textual properties of the workspace. The results of the implicit type symbol analysis and the output of the spatial parser are used to identify the candidate visual structures. Candidate structures must contain elements of the same object type, or implicit type if the objects are untyped. Once the candidate locations are identified, textual similarity based on term vectors of the pairs of structures are used to determine the best pairing. The suggestion will only be made if the textual similarity is above a particular threshold.

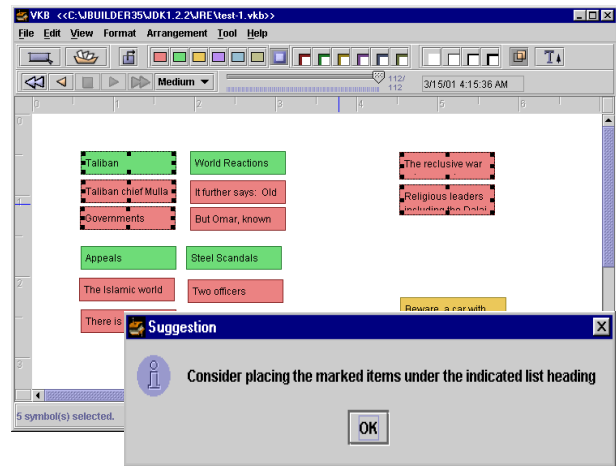


Figure 6: A placement selection is made based on visual and textual information from the workspace.

The following scenario demonstrates the placement suggestion agent. The information workspace for this scenario is shown in Figure 6. (Note that the suggestion window would only appear after requested by the user.) A visual structure consisting of four labeled lists on the upper left hand side of the workspace contains data collected regarding the rise of the Taliban in Afghanistan. The lists have a green list label and red list symbols. The composite at the right hand bottom, consisting of a single orange symbol contains information regarding election scandals in India.

As shown in the figure, the user has created an unlabeled list of two red colored symbols containing data regarding fundamentalism in Afghanistan. In this context, the workspace has four candidate labeled lists based on visual similarity. Based on the text analysis, one of the lists is selected as similar enough to consider as a destination for moving the new elements.

The example structure is somewhat simple. The benefits would be greater in a larger information workspace with a large number of candidate labeled lists.

### 6.2.4 Relationship Suggestions

In VKB, users can define explicit relationships between symbols. In many such relationships, a symbol in one composite may be associated with a symbol in a different composite. When the user identifies these explicit relationships between symbols in different composites, there may exist several such relationships that have yet to be explicitly identified. Hence, analogical reasoning can be used to identify and suggest these relationships to the user.

The suggestion algorithm uses the results of the spatial parser to identify candidate source-destination clusters based on existing explicit relationships. Then, textual similarity between the symbols in these candidate cluster pairs is measured and new relationships are suggested if textually similar pairs of symbols are found. The following scenario provides an example.

Consider a workspace similar to that shown in Figure 6. In this case, the user is collecting information on the rise of the Taliban in Afghanistan, on corruption in India, and on Muslim culture. As the user begins to create relations between the separate lists, the system looks for analogous relations -- i.e. potential relations between symbols within similar structures. Such suggestions can help the user identify relations they have overlooked or act as shortcuts for creating relations the user plans to create later.

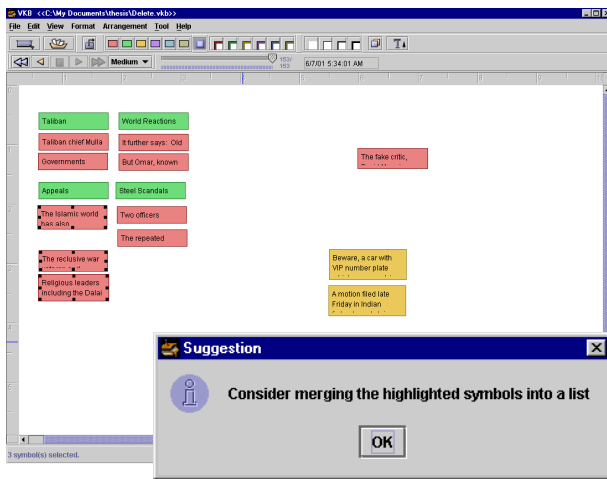


Figure 7: A suggestion is made to close the gap in the list structure based on visual, textual, and temporal information.

### 6.2.5 Cleanup/Consolidation Suggestions

During the course of working in a spatial hypertext, the user modifies the visual workspace by creating, editing and deleting symbols. This can result in existing structures becoming more difficult to perceive and new structures being recognized in their place. In such situations, suggestions that identify likely misinterpretations and help clean up the workspace to better reflect the intended structure would help users manage their information space. The following scenario, based on the layout in Figure 7, concerns the gap in a list created when the user removed one element.

To recognize such situations, the system goes back in history searching for edit events (e.g. deletion of a symbol) in the workspace. It then compares the structure of the workspace before the edit event with the current structure. When the edit caused an existing structure to break apart, it is a candidate for cleanup. But the deletion of a symbol from a composite (or other edit event) might have intentionally divided the original composite into two composites. On the other hand, the deletion might have created gaps in the composite that visually gives the impression of there being two separate composites. To determine which of the above cases is true, the system considers the textual similarity between the two composites. If the textual similarity is high, the system suggests the merging of the two composites. This algorithm utilizes spatial, textual and temporal data about the workspace.

## 7. USER EVALUATION

The long-term goal is that the suggestion-based proactive support added to VKB help users engaged in real knowledge building tasks. The current efforts are the first components towards achieving that goal. As such, we evaluated the suggestion manager and some of the suggestion agents individually. The next subsections briefly describe a comparison between suggestion delivery in the VKB Suggestion Manager and the Microsoft Office Assistant and an evaluation of the usefulness and quality of suggestions being generated.

### 7.1 VKB Suggestion Manager and the Microsoft Office Assistant

Microsoft implemented a suggestion mechanism in its Office Assistant. The “paperclip,” as it is often called, had mixed reviews. However, many truly loathed the character [28].

Moreover, Microsoft is currently using the fact that the Office Assistant is not the default option for Office XP as a selling point for the product [24]. Given such experience, are all suggestion interfaces going to incur similar responses? To answer this question, we asked users (students not associated with VKB) to compare the VKB and Office Assistant notification systems.

Users were asked to compare the notification system used by Microsoft Word's Office Assistant with the system used in VKB. Users completed simple tasks in both systems designed to cause some type of suggestion to be generated. Users were then asked to answer a set of questions for both systems.

The questions asked, average response scores, and p-values for seven participants are shown in Table 1. Each question applied to both the Office Assistant and the VKB suggestion manager. The participants responded used a Lickert scale from 1 (strongly agree) to 5 (strongly disagree). Question 2 had statistical significance at  $\alpha = 0.05$ , question 7 at  $\alpha = 0.10$ , questions 3 and 4 at  $\alpha = 0.20$ . Bold values show the better value -- this is the lower value for questions 1, 3, 4, 5, and 6 and the higher value for questions 2, 7, and 8.

Question	OA	VKB	P
1. Suggestion/help notification is effective.	<b>2.43</b>	2.57	0.74
2. The method of notification disturbs my work.	2.14	<b>4.00</b>	0.04
3. I can address suggestions/help when I wanted.	2.71	<b>1.71</b>	0.11
4. I feel I received all suggestions/help generated.	<b>2.00</b>	2.86	0.11
5. I can access all suggestions/help generated.	2.57	<b>1.57</b>	0.23
6. Animation was appropriate.	2.43	2.43	1.00
7. Animation was distracting.	2.43	<b>3.86</b>	0.06
8. Animation caught my attention when there was no suggestion/help.	2.43	<b>3.00</b>	0.39

Table 1: Evaluation results for VKB and MS Office Assistant

One area where VKB significantly improved on the Office Assistant was in the area of distraction. Question 2 indicated that the notification method does not affect users' work. Question 7 indicated that users find the animation in the Office Assistant more distracting than the animation in VKB.

The second area of difference, seen in the answers to questions 3 and 5, is enhanced user control over suggestions. VKB users felt they could get to all the suggestions generated when they wanted. Slightly in contrast to this, though, users had a greater belief that they received all suggestions in the Office Assistant (question 4).

In general, the answers show that VKB's approach to notification is no less effective than the established system in the Office Assistant and shows some distinct advantages. A caveat is that users responses may have been biased by experiences with prior versions of the Office Assistant or by its reputation.

### 7.2 VKB Suggestion Agents

Informal evaluations of the Placement, Inferred Relationship and Cleanup/Consolidation suggestion types were carried out. Four users were asked to evaluate the various suggestion types. The primary purpose of the evaluation was to determine the usefulness of a suggestion type. A secondary goal was to determine the quality of the particular suggestion algorithm by comparing generated suggestions to user decisions.

### 7.2.1 Evaluating Usefulness

For each suggestion type, a VKB workspace was created as a starting point upon which the subject would continue collecting and organizing information. The users were given 15 minutes per scenario to collect information from the web and add it to the workspace. When a suggestion was triggered, the users were asked to rate the usefulness of that particular suggestion. The ratings were: *very useful*, *somewhat useful*, and *not useful*.

**Placement Suggestions.** The placement suggestion type was evaluated for placement of both a single symbol and a list of symbols. All four subjects rated suggestions for placing a single symbol as *very useful*. Two subjects rated the structure placement suggestions as *very useful* and two rated them *somewhat useful*. In general, subjects appreciated the utility of placement suggestions, especially in a very large workspace where it is easy for a user to lose track of information. The users were of the opinion that the system suggesting the best location to place newly collected information would reduce their work.

**Relationship Suggestions.** Subjects were given a workspace with existing relationships and more were suggested as they added information to the workspace. Suggestions for relationships between individual elements were unanimously rated *somewhat useful*. Relation suggestions between higher-level structures were rated as *not useful* by three of the four subjects, with one *somewhat useful* rating. Subjects' comments indicate that they did not perceive a value for VKB relations since navigation was the only feature impacted by their existence. They agreed that in a very large workspace, with a lot of relation links connecting multiple composites, such suggestions might be useful.

### 7.2.2 Evaluating Quality

Users were shown a pre-scripted scenario of use for a particular suggestion type in a VKB workspace. They were then asked the best suggestion available in the current scenario for the suggestion type. Their input was then compared with the suggestion generated by VKB. Note that this is a very tenuous notion of quality as it only compares the first VKB-generated suggestion to the first suggestion made by subjects. This would tend to cause the comparison to be overly critical. On the other hand, the scenarios were simpler than real knowledge building tasks so the chances of agreement were higher than would normally be expected.

**Placement Suggestions.** Again, the suggestion type was evaluated for placement of both a single symbol and a list of symbols. The VKB generated suggestion for placing a single symbol matched the decision of three of the four subjects. In the task of placing a list of symbols, the VKB-generated suggestion matched the unanimous decision of the subjects.

**Relationship Suggestions.** Users were provided a workspace with a number of existing visual structures with some existing relations between elements in these structures. They were asked to suggest one more relation between symbols and one between higher level structures. The VKB-generated symbol-to-symbol suggestion matched the choice of three of the four subjects. In the case of the structure to structure relation, the VKB-generated suggestion matched that chosen by two of the four subjects.

**Cleanup/Consolidation Suggestions.** Subjects were given a VKB workspace containing a list in which one element had previously been deleted. The user is asked whether the symbols above and below the deleted symbol still constitute a list and should be merged and gap eliminated. All the users suggested that the lists should be merged. This matched the output of VKB.

## 8. DISCUSSION & CONCLUSIONS

With growing effort surrounding standards for representing and sharing knowledge via the semantic web, it is important to develop tools that will aid in the expression of knowledge. The formal expression of semantic content is difficult due to the tacit and situated nature of much human knowledge. Given these difficulties, knowledge expression cannot be viewed as the encoding of existing knowledge but as a constructive process that changes the knowledge in the human while creating the external representation.

Spatial hypertext is well suited for such a knowledge building process as it allows the initial expression of content to take informal, even ephemeral, forms. The system can proactively support the incremental formalization of this content by recognizing patterns within the informal content and providing suggestions based on these patterns.

VKB's proactive support for knowledge building is through its suggestion manager and suggestion agents. The suggestion agents generate suggestions based on the analysis of visual, textual, and temporal information in the information space. Combining these characteristics of the workspace enables the generation of suggestions not possible when using only one type of analysis.

Trade-offs exist among effective notification, distraction, trust, and control when designing suggestion-based assistants. The VKB suggestion manager attempts to balance these goals by using an unintrusive notification animation, progressively disclosing suggestion details, making suggestions persistent and accessible through a suggestion history, and providing user control through interfaces for tailoring both the generation and notification of suggestions. A comparison with the MS Office Assistant indicates that the effectiveness of the VKB suggestion manager is similar while being less distracting and allowing users to consider suggestions when they want.

The informal evaluation of VKB suggestion agents shows that, in simple situations, their suggestions match the decisions of people. Further evaluation within "real world" tasks is clearly needed. The evaluation also indicates that suggestions for placement of new or lost elements that aid in the informal expression of relations were viewed as quite useful. In contrast, the applicability of suggestions for formally-represented relations was not obvious. The primary reason for this is that there must be a benefit to formalizing content and the tasks users were performing did not provide such motivation. In the end, knowledge building will not happen until a benefit is perceived by those who must put forth the extra effort, regardless of standards for formal knowledge representation or system support for formalization.

## 9. ACKNOWLEDGMENTS

This work was supported in part by grant IIS 9734167 from the National Science Foundation.

## 10. REFERENCES

- [1] Akscyn, R.M., McCracken, D.L., and Yoder, E.A. KMS: A Distributed Hypermedia System for Managing Knowledge in Organizations. *Communications of the ACM*, 31, 7, 820-835.
- [2] Bernstein, M. An Apprentice That Discovers Hypertext Links. *Proceedings of the 1990 European Conference on Hypertext*. 1990, pp. 212-223.
- [3] Carlson, D., and Ram, S. HyperIntelligence: The Next Frontier. *Communications of the ACM*, 33, 3, 311-321.

- [4] Conklin, J., and Begeman, M. gIBIS: A Hypertext Tool for Exploratory Policy Discussion. *Proceedings of CSCW'88*, ACM, New York, 1988, pp. 140-152.
- [5] Davis, R. Interactive Transfer of Expertise. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, B.G. Buchanan, E.H. Shortliffe, Eds. Addison-Wesley, Reading, Mass., 1984, pp. 171-205.
- [6] Fischer, G. User Modeling in Human-Computer Interaction, *User Modeling and User-Adapted Interaction*, 11, 1/2 (2001), pp 65-86.
- [7] Fischer, T., Lemke, A.C., & Schwab, T. Knowledge Based Help Systems. *Proceedings of CHI '85*, ACM Press, 161-167.
- [8] Frank, M. and Foley, J. A Pure Reasoning Engine for Programming By Demonstration. *Proceedings of UIST'94*, ACM, New York, Nov 1994.
- [9] Frank, M., Sukavirija, P., and Foley, J. D., Inference bear: designing interactive interfaces through before and after snapshots. *Proceedings of the ACM Symposium on Designing Interactive Systems*, ACM, 1995, pp. 167-175.
- [10] Furnas, G.W., Landauer, T.K., Gomez, L.M., and Dumais, S.T. The Vocabulary Problem in Human-System Communication. *Communications of the ACM* 30, 11 (Nov. 1987), pp. 964-971.
- [11] Hofmann, M., Schreieis, U., and Langendörfer, H. An Integrated Approach of Knowledge Acquisition by the Hypertext System CONCORDE. *Proceedings of the 1990 European Conference on Hypertext*. 1990, pp. 166-179.
- [12] Hollan, J., Rich, E., Hill, W., Wroblewski, D., Wilner, W., Wittenburg, K., and Grudin, J. An Introduction to HITS: Human Interface Tool Suite. In *Intelligent User Interfaces*, J. Sullivan, S. Tyler, Eds. ACM, New York, 1991, pp. 293-338.
- [13] Horvitz, E. Principles of Mixed-Initiative User Interfaces. *Proceedings of CHI '99*. ACM Press, 159-166.
- [14] Igarashi, T., Matsuoka, S., Masui, T. Adaptive recognition of implicit structures in human-organized layouts. In *Proceedings, 11th IEEE International Symposium on Visual Languages*, 1995, pages 258-266
- [15] Jordan, D.S., Russell, D.M., Jensen, A.-M.S., and Rogers, R.A. Facilitating the Development of Representations in Hypertext with IDE. *Proceedings of Hypertext '89*. ACM, New York, 1989, pp. 93-104.
- [16] Kaindl, H., and Snaprud, M. Hypertext and Structured Object Representation: A Unifying View. *Proceedings of Hypertext '91*. ACM, New York, 1991, pp. 345-358.
- [17] Malone, T.W., Grant, K.R., Lai, K.-Y., Rao, R., and Rosenblitt, D. Semi-Structured Messages are Surprisingly Useful for Computer-Supported Coordination. *Proceedings of CSCW'86*. 1986, pp. 102-114.
- [18] Marshall, C.C. Exploring Representation Problems Using Hypertext. *Proceedings of Hypertext '87*. ACM, New York, 1987, pp. 253-268.
- [19] Marshall, C., Halasz, F., Rogers, R., and Janssen, W. Aquanet: a hypertext tool to hold your knowledge in place. *Proceedings of Hypertext '91*. ACM, New York, 1991, pp. 261-275.
- [20] Marshall C., and Shipman, F. Spatial hypertext: designing for change. *Communications of the ACM*, 38, 8, 88-97.
- [21] Marshall C.C., and Shipman, F.M. Effects of Hypertext Technology on the Practice of Information Triage. *Proceedings of ACM Hypertext '97 Conference*, 1997, pp. 167-176.
- [22] McCall, R., Mistrik, I., and Schuler, W. An Integrated Information and Communication System for Problem Solving. *Proceedings of the Seventh International CODATA Conference*. Pergamon, London, 1981.
- [23] McCall, R., Bennett, P., d'Oronzio, P., Ostwald, J., Shipman, F., and Wallace, N. PHIDIAS: Integrating CAD Graphics into Dynamic Hypertext. *Proceedings of the 1990 European Conference on Hypertext*. 1990, pp. 152-165.
- [24] Microsoft website with Microsoft XP and Clippy: <http://www.microsoft.com/Office/clippy/>
- [25] Nanard, J., and Nanard, M. Using Structured Types to incorporate Knowledge in Hypertext. *Proceedings of Hypertext '91*. ACM, New York, 1991, pp. 329-343.
- [26] Peper, G., MacIntyre, C., and Keenan, J. Hypertext: A New Approach for Implementing an Expert System. *Proceedings of 1989 ITL Expert Systems Conference*, 1989.
- [27] Polanyi, M. *The Tacit Dimension*, Doubleday, Garden City, NY, 1966.
- [28] Quistgaard, K. Kill Mister Paperclip! 2000. Available at: <http://www.salon.com/tech/log/2000/06/22/paperclip/>
- [29] Randall, N. and Pederson, I. Who Exactly is Trying to Help Us? The Ethos of Help Systems in Popular Computer Applications, *Proceedings of the sixteenth annual international conference on Computer documentation*, 1998, pp. 63 - 69.
- [30] Schön, D. *The Reflective Practitioner: How Professionals Think in Action*. Basic Books, New York, 1983.
- [31] Schwabe, D., Feijo, B., and Krause, W. Intelligent Hypertext for Normative Knowledge in Engineering. *Proceedings of the 1990 European Conference on Hypertext*. 1990, pp. 123-136.
- [32] Shipman, F., Hsieh, H., Airhart, R., Maloor, P., Moore, J.M., and Shah, D. Emergent Structure in Analytic Workspaces: Design and Use of the Visual Knowledge Builder. *Proceedings of Interact 2001*, 2001, pp. 132-139.
- [33] Shipman, F., Hsieh, H., Airhart, R., Maloor, P., and Moore, J.M. "The Visual Knowledge Builder: A Second Generation Spatial Hypertext", *Proceedings of the ACM Conference on Hypertext*, 2001, pp. 113-122.
- [34] Shipman, F.M. and Marshall, C.C. Formality Considered Harmful. *Computer Supported Cooperative Work (CSCW)*, 8, 4 (Fall 1999), pp. 333-352.
- [35] Shipman, F., Marshall, C., and Moran, T. (1995), Finding and using implicit structure in human-organized spatial layouts of information. *Proceedings of CHI '95*, ACM, 346-353.
- [36] Shipman, F. and McCall, R. Supporting Incremental Formalization with the Hyper-Object Substrate. *ACM Transactions on Information Systems*, April 1999, pages 199-227.
- [37] Smolensky, P., Bell, B., Fox, B., King, R., and Lewis, C. Constraint-Based Hypertext for Argumentation. *Proceedings of ACM Hypertext '87*, 1987, pp. 215-245.
- [38] Suchman, L.A. *Plans and Situated Actions: The problem of human-machine communication*. Cambridge University Press, Cambridge, UK, 1987.
- [39] Thomas C., Fischer, G. Using Agents to Personalize the Web. *Proceedings of the 1997 International Conference on Intelligent User Interfaces*, ACM, 53-60.
- [40] Turney, P. D. *Extraction of Key phrases from Text: Evaluation of Four Algorithms*. NRC Technical Report ERB-1051, 1997.