# Practice Based Adaptive Systems

Joshua Introne and Richard Alterman
Volen Center for Complex Systems
Brandeis University
[jintrone, alterman@cs.brandeis.edu]

## Abstract

A critical task in building adaptive systems is finding the right AI technology to infer users' needs and preferences. But the power of an adaptive system also critically depends on choosing the right set of operators and representations at the interface. This is because the representational system employed at the interface ultimately determines the type of information that will be available to drive back-end adaptive system.

We describe a methodology for adding adaptive functionality to *groupware* systems. Our focus is on how the addition of interface tools (called Coordinating Representations) can both support user efforts to stay coordinated and simplify user intent inference at runtime. In using Coordinating Representations to stay coordinated, users produce a stream of structured information that is highly relevant to intent inference. This in turn supports the introduction of powerful adaptations using standard AI techniques. Our approach is applicable to all computer-mediated collaborative tasks.

We will describe the application of this approach to an existing groupware system. An earlier study validated our methodology for developing Coordinating Representations that improved users' performance in the domain task of our testbed. The current study validates that the information derived from the Coordinating Representations can be leveraged to produce an effective adaptive user component.

# 1 Introduction

The computer is an artifact that mediates the interaction between a subject (the user) and object (the domain) (Norman, 1991). Unlike most tools, computers are largely domain independent; with the addition of appropriate software, sensors and effectors, computers can support user interactions with a wide variety of domains (including other users). This flexibility is possible because computers introduce a layer of symbolic abstraction between the user and domain. This layer of abstraction maps the user's input to some activity or state change in the domain, and information about domain's state back into some symbolic representation for the user. The power of the approach derives from the fact that this mapping can be chosen in ways that simplify complicated or error prone domain actions, and clarify and augment important information. Of course, it is also possible to create mappings that constrain or disrupt activity, and obscure or misrepresent information.

The design of any software system thus involves choosing an interface language, a domain model, and a mapping to translate between the two. As it is not generally possible to capture a set of operations that are ideal for all users, most systems today are *adaptable* (Oppermann, 1994); in addition to domain specific operators, adaptable systems provide a set of meta-operators that allow users to customize the behavior of the application. While such flexibility is useful, it places the burden of customization entirely upon the user. The operators are frequently closer to the procedural language of the system itself, requiring the user learn a rudimentary programming language in order to customize the system. The cost of learning to use adaptable features is often too high for ordinary users.

*Adaptive* systems are designed to address this problem. They offload user work or provide guidance to help the user achieve their goals, ideally without introducing additional work. An adaptive mechanism introduces a more complicated and / or possibly dynamic mapping from interface to domain model. An adaptive system might allow a user to express their intentions in a high-level language, which is mapped by the system to lower level directives in the procedural language of the system, but this introduces an additional special-purpose language to the interface that may distract the user from the domain task. Alternatively, an adaptive component might use interface actions directly to infer higher-level domain intent and use these inferences to offload user work; however, existing interfaces frequently do not capture enough information of the right granularity to support useful intent inference.

We will refer to the set of operations and representations that are available to the user in the interface as an *interface language*. In adaptive systems, the interface language constrains both the usability of the system and the power of the adaptive component. This is because the interface language determines how much and what kind of information is available to the system for the adaptive component, as well as how much work the user has to do to use the system. In the following, we will examine this tradeoff in various implemented approaches. We will then present a subset of interface languages that both improve the interface from the user's perspective and make it easier for the developer to add adaptive capabilities to a system. This methodology involves adding specially designed representations that mediate the user's online interaction to an existing groupware system in order better support coordination. These tools, in turn, accumulate usage information in a form that simplifies intent inferencing. We call these interface representations Coordinating Representations (CRs).

In an earlier study (Alterman et al., 2001a), we showed that users adopt appropriately designed CRs, that these CRs improve users' task performance, and that they reduce communication in other modalities (textual chat). In this work, we evaluate an adaptive component that uses runtime data from one CR to support user intent inference, presenting a study that shows that this adaptive component was adopted by users, while reducing coordination errors users' cognitive load.

The remainder of the paper is organized as follows. In Section 2, we provide background to frame our approach, focusing on how the tradeoff between user work and inferential power is dealt with in other adaptive systems. In Section 3, we describe how CR's support coordination, and provide a brief overview of our methodology. This methodology is presented in more detail in other papers (Alterman, et al, 2001a; Feinman and Alterman, 2003), which empirically show how CR's improve user performance along several dimensions. In Section 4, we introduce our test domain (experimental platform), and examine data to show why extracting reference information that is relevant to the intent inference problem is very hard. We will then show how CR's make this information more readily accessible to the system at runtime. In Section 5, we present our technical approach and validation study. First, we present a Bayesian Belief Network (BN) -- that was developed quickly and easily using off-the-shelf software -- to infer user intentions in our test domain. Then, we describe our adaptive component, which aids user plan construction in our test domain. Finally, we present details from a human subject experiment that confirms the utility and effectiveness of the adaptive component. In Section 6, we will discuss the implications of our work, and conclude.

# 2 Interface Languages and Adaptive Systems: Common Approaches

We identify two major classes of interface languages that have previously been employed in developing adaptive systems; task specific languages and intent expression languages. We will examine several implemented systems with respect to this classification, and focus on the usability and engineering tradeoffs implicit in the choice of interface language.

### 2.1.1 Task specific languages

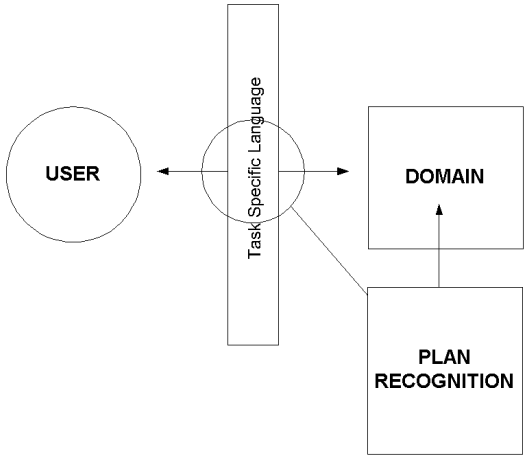



**Figure 1: Plan recognition with task specific languages**

An interface language that is task specific is designed to facilitate direct user action in the domain (see Figure 1). A task-specific language mediates the user's interaction with a domain. The majority of software systems employ task specific interfaces, but such interfaces do not generally capture enough information about the user's activity to support task-level adaptation.

One system that does employ a task-specific interface and provides adaptive user support is the AIDE system for data exploration, described by (St. Amant and Cohen, 1998). AIDE is a direct manipulation system, in which the user and system share a common representation of the complete state of the problem via a graphical and tabular display of data. The user can select and manipulate this representation to perform various statistical operations. The adaptive component assists the user during the natural course

of the user's exploration of the data by inferring higher-level goals from the user's interface actions. For example, if a user requests a regression on a selected set of data, the system will run several different regressions to find the best fit, and identify outliers for the user. Control strategies encoded in the process provide sufficient opportunity for user-direction at each decision point in the exploration of the data.

The interface to the AIDE system has two properties that are frequently found in direct manipulation systems (Hutchins, Hollan, and Norman, 1986) and make adaptation possible:

- There is a high degree of similarity between the tasks represented by operators in the interface and the tasks in the domain. A high degree of operator-task similarity can make the interface language easier for the user to learn. (Norman, 1991; Schneiderman, 1983) It can also make it easier for the system to extract domain intentions from user activities. (St Amant and Young, 2001).
- Complete information about the domain is shared between the user and the system. The degree to which the system can represent the information that is relevant to the user's domain task ultimately determines the scope of adaptation. (Terveen, 1994; Terveen et al., 1995).

Because the interface is giving us direct information about the tasks the user is trying to achieve, and information about the domain with respect to the user's task is complete, it is possible to add an adaptive component that will perform useful task-level adaptation without significant modifications to the interface language. As demonstrated by AIDE, no additional work is created for the user to reap the benefits of the adaptive component.

Not all domains support direct manipulation (Schneiderman, 1998), and not all direct manipulation interfaces exhibit the two properties described above. For example, document editing applications do not maintain a deep representation of the semantic content of unstructured text. Thus, adaptation is restricted to the discovery of macro-operators at the level of syntactic and structural manipulation (e.g. Cypher, 1991; Miller, 2002): An adaptive component might recognize that a user wants to indent each paragraph upon observing the user has indented several. However, an adaptive component would not be able to recognize that a user wants to italicize words that introduce new concepts throughout a document, regardless of the user's activity, because the system does not maintain a representation of a document at a conceptual level.

### 2.1.2 Intent Expression Languages

In cases where a task-specific language does not provide enough information to support adaptation, an *intent expression language* is usually provided in addition to an existing task specific language (Figure 2). An intent expression language is designed to facilitate the user's expression of intent to the system.

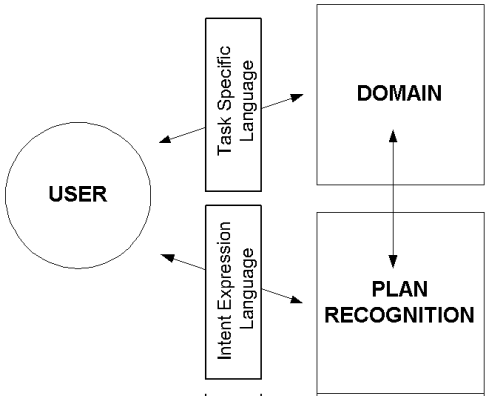



**Figure 2: Plan recognition with intent expression languages**

The most ambitious approaches to developing intent expression languages employ natural language; most of these use spoken language (see McTear, 2002 for a review). Spoken language systems have many potential benefits; in particular, they could make computers accessible to a wider range of people, and spoken language interfaces may be employed where traditional interfaces are ineffectual.

Currently, spoken language interfaces are only possible for limited and well-structured domains, such as trip planning or email systems (TRIPS, Ferguson and Allen, 1998; ELVIS, Walker, et al. 1998). It is unlikely that spoken language interfaces will be available for richer domains in the foreseeable future because of the limitations of natural language processing technology (Allen and Ferguson, 2002). For the highly structured domains for which spoken languages can be built, more highly constrained interfaces may be preferable (Nardi, 1993): at a concrete level, it may simply be easier to drag an icon on a direct manipulation interface than to verbalize a goal to plan a leg of a journey from point A to point B.

Another feature of spoken language interfaces is that they do not (and cannot) constrain the user's input. This leads to two problems. First, from the user's perspective, the interface appears to support a wider range of conversational moves and vocabulary than the back-end system can cope with. The user needs to learn what subset of natural language the system actually supports, and this can lead to user frustration and error (Schneiderman, 1998). Second, from the engineer's perspective, input to the back-end plan-recognition system cannot be guaranteed to be correct, which further complicates the design of the system.

Highly constrained, formal intent expression languages can fix some of the problems with natural language. An example of this approach is the Collagen system (Rich and Sidner, 1998; Lesh, et al., 1999). Collagen is a middleware framework for the development of collaborative agents that can provide guidance and offload some user work in the domain task. Collagen includes a formal, domain-independent intent expression language and a general planning component designed to work with the intent language. Collagen was first designed to support trip planning in a direct manipulation interface. The initial system did not employ plan-recognition, but did introduce a formal language for intent expression called ADL (Artificial Discourse Language) (Sidner, 1994), so that the system could respond to the user's task-level intent.

ADL is implemented as a dialog driven language, which appears in the interface as human readable (English) phrases. The interface restricts the conversational moves that are available to the user by presenting a list of options from which the user must select, one at each turn in the dialogue. Internally, the ADL is a predicate calculus that provides several operators supporting intent expression and user queries, which maps directly to the plan representation used by the system. To minimize the amount of work users need to do to explain their actions to the system using ADL, Collagen (Lesh et al., 1999) now incorporates a plan-recognition component to reduce the amount of interaction a user has to have with the agent.

Collagen solves an engineering problem that complicates natural language systems by constraining the intent expression language so that user input to the back-end plan recognition system is correct. It also eliminates the need for the user to learn which dialogue moves are possible at each point in a conversation. However, the interface language is cumbersome for users; as noted by Collagen's developers, "it is often more efficient and natural to convey intentions by performing actions" (Lesh et al., 1999; page 1). The addition of a separate intent expression language also requires the user to switch focus between the domain task and interaction with the adaptive component, which can lead to breakdowns and user frustration (Bödker, 1991).

As discussed above, the lack of constraints on user input in natural language interfaces creates both engineering and usability problems. On the other hand, more formal languages that include constraints on dialogue moves are cumbersome and inefficient from the user's perspective. An approach that attempts to find a workable compromise is to constrain the vocabulary of the intent expression language, but relax constraints on the dialogue moves available to the user.

An example of this third approach is the integration effort between ForMAT and the Prodigy/Analogy planner (Cox and Veloso, 1997a). ForMAT was an existing case-based plan retrieval tool used by military planners for the retrieval of past force deployment planning episodes. ForMAT provided a task specific language via which users would specify goals and constraints to select plans from a case base, and then modify the retrieved plans to fit their current needs. The adaptive component was included to help with the modification of retrieved plans. It was driven by the Prodigy/Analogy plan generation and retrieval system, a general planner that lacked a non-technical front-end.

The ForMAT language was extended with a mechanism for users to describe their goals to the planner, and to retrieve output from the Prodigy planner. These operators were designed to support conversation-like mixed initiative planning. However, it was found that,

> "Normally the goals given to a planning system by a knowledgeable and sympathetic user are in a well-structured format. But goals provided by a more realistic less restrained user present at least three problems to traditional planning systems: (1) *input goals may actually be specified as actions rather than state conditions;* (2) *they may be abstract rather than grounded;* and (3) *they may include subgoals along with top-level goals.*" (Cox & Veloso, 1997b; page 1)

It was necessary to incorporate pre-processing and add additional heuristic control strategies to the Prodigy/Analogy system to accommodate these behaviors, and it is not clear how effective these strategies were in practice. Subsequent efforts by the authors have abandoned ForMAT in favor of an entirely different interaction metaphor (goal-manipulation rather than search) (Cox, 2003), which is specific to and constrained by the type of tasks Prodigy/Analogy is good at.

The ForMAT / Prodigy integration effort illustrates the potential engineering difficulties that may be encountered where an intent expression language does not constrain user input to match the requirements of a traditional, "batch-processing" plan based adaptive system, even though the vocabulary may be reduced. The differences between an intent expression language modeled after human conversation and the input language for traditional AI planners are too great for a simple mapping from one to the other to suffice.

## 2.2 Summary

We have reviewed several common approaches to building adaptive, plan-based systems according to the kind of interface language employed in each case. We have presented two major classes of interface languages; task specific and intent expression. For some types of task specific languages, such as direct-manipulation systems, it is possible to add task-level adaptive support without modifying the interface language. Direct-manipulation interfaces have also been shown to easy to use, powerful, and enjoyed by users (Schneiderman, 1993).

However, not all task-domains can support direct-manipulation, often because it is not possible to represent all of the information about the task-domain that is relevant to a user's task level behavior. In these cases, a common solution is to add an intent expression language to solicit task level intent information from the user directly. In interfaces where an intent language co-exists with a task specific language

(which is the majority of cases), there are some usability concerns because users must learn two languages, and divide their attention between the domain task and intent expression task.

We have described three common configurations for intent expression languages and the attached adaptive component. One approach is to employ intent expression languages that are as close as possible to natural language. To date, these systems can only be employed in highly structured, and limited domains. For such domains, natural language interfaces present two problems; 1) they do not reflect the limitations of the back-end plan-recognition system, and therefore present the user with a difficult learning task; and, 2) they cannot limit the user's input to be correct, and thus present a difficult engineering problem. An alternative approach, designed to alleviate the engineering difficulties of natural language interfaces, employs languages that constrain the user's input so it can be easily mapped to the system's representation. Although such languages simplify engineering issues, they are cumbersome for users. Finally, a third approach employs a restricted vocabulary that is more loosely constrained. However, even with a restricted vocabulary, an unrestrained user will specify intentions in ways that cannot be interpreted correctly by well-established AI planning techniques.

# 3 The Interface Language Redux

We present an alternative approach to developing an interface language that can support adaptation. Our approach is based on an enriched notion of the human-computer interface as a representational system. In this analysis, the representational system serves as more than just a conduit for information exchange. This notion is based upon the study of representations and representational systems that a community has available in the workplace (Hutchins 1995a, 1995b; Norman 1991). A representational system has three parts:

1. A set of representational media available to the participants.
2. A set of internal or external, private or shared, representations, including those provided in the design of the task environment and ones created at runtime.
3. A set of procedures for communicating, recording, modifying, transcribing, and aligning multiple, partial representations of the shared context.

We refer to artifacts in a representational system that people use to stay coordinated in an ongoing cooperative activity as *coordinating representations* (CRs) (e.g. Alterman, et al. 2001b; c.f., Suchman & Trigg, 1991; Goodwin & Goodwin, 1996; Hutchins 1995a, 1995b). CRs are ubiquitous in the world. For example, the stop sign is a type of coordinating representation that signifies an organization for an expected behavior. It presents (but does not enforce) a structure for organizing the collective behavior of drivers, pedestrians, and cyclists at a busy intersection. In our approach, we add CRs to the representational system in an existing groupware system to improve coordination in the maintenance of common ground. In this paper, we leverage the information users place in the CRs to provide adaptive support.

## 3.1 Coordinating Representations and Groupware

Groupware systems support groups of people engaged in a common task (or goal). They provide an interface to shared environments. They facilitate communication, coordination, and collaboration of group effort. (Ellis, Gibbs, and Rein, 1991) The construction of a groupware system requires an analysis of the work environment in which it will be used and a design of a mediated interaction among the users. Much of the work for the users is to coordinate their actions and talk. A significant constraint on coordination is that their joint behavior is mediated, and consequently communication lacks some of the immediacy and effectiveness of face-to-face activities. During the construction of the groupware system,

methods that support the interaction among the users are included into the design of the system. For synchronous activities the virtual shared whiteboard and textual chat are examples of general-purpose methods that support interaction. (Shneiderman, 1998)

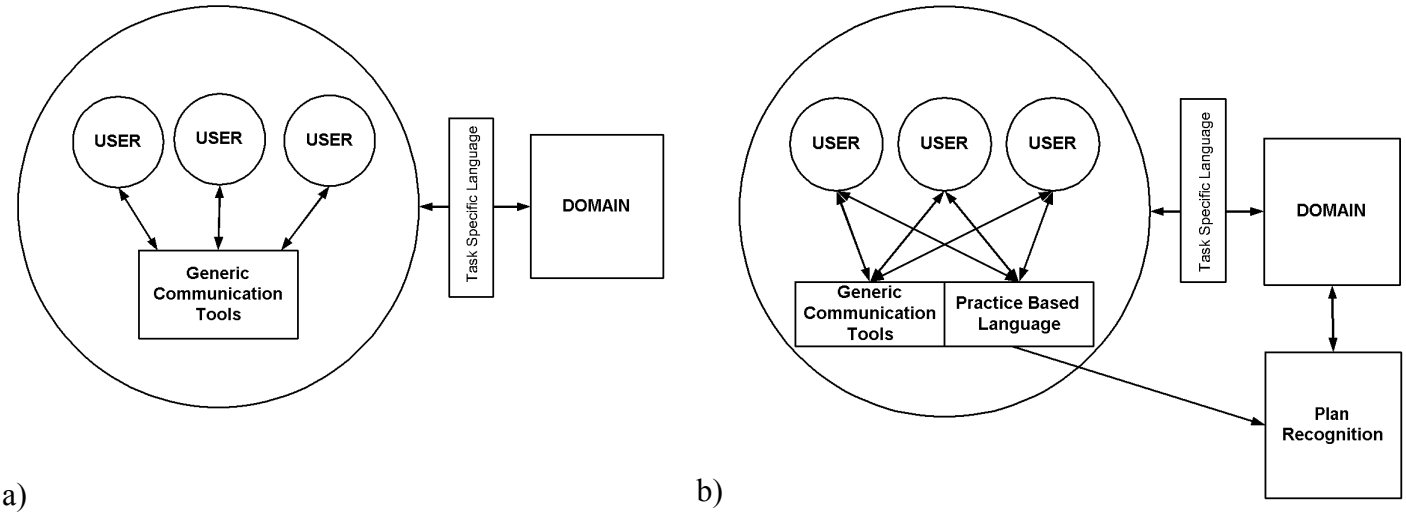
Unlike general-purpose coordination tools in a groupware system, Coordinating Representations support specific types of information. CRs lend structure to this information that makes it easy for people to use. People communicate about the domain through and in reference to a CR (Suchman and Trigg, 1991), and it becomes an integral part of the conversation that allows coordination to take place. An appropriately designed CR will improve the users' performance in the domain task.

The type and structure of the information captured by a CR enables the introduction of autonomous algorithms for task level adaptation. A CR captures state information that may be missing, or is only available in a form that is difficult for a computer to interpret. Because users organize their behavior in ways that are more efficient and less error prone, it will be easier to interpret via standard plan recognition techniques. Finally, because a CR is an integral part of the coordination practice, the information in a CR will be up to date and relevant. We distill these observations into three design criteria for CR's that will support adaptation in groupware systems;

1. A CR should introduce work that users want to do, so that they adopt the CR.
2. Users should be able to organize their task in ways around a CR that improve their performance.
3. The CR should impart structure to coordinating information that can be leveraged by traditional AI methods to infer intent.

In the following section we provide an overview of a methodology that leads to the development of CRs that meet these criteria. This methodology is described in more detail in Alterman (2001a) and Feinman and Alterman (2003).

## 3.2 Methodology




a) b)

**Figure 3: a) Standard groupware system; b) Groupware system with practice based language that supports plan recognition**

Our methodology begins with an existing online practice that is grounded in the representational system provided by a groupware system (e.g. chat, a shared whiteboard) (Figure 3a). Our methodology then proceeds as follows:

1. Transcripts are collected of online user behavior.
2. These transcripts are analyzed in order to identify weak spots in the representational system.
3. The analysis of practice is used to improve the online exchange and management of information by adding coordinating representations that improve coordination in the maintenance of common ground.
4. **Information collected by the CRs is leveraged to drive task-level adaptive components.**

Our approach to data collection and analysis has roots in ethnography and interaction analysis (Suchman & Trigg, 1991). Traditionally, ethnography has used "in situ" observation and videotape technology to capture and analyze naturally occurring workplace activity. For same-time different place groupware systems, all coordination work is mediated by the system and can be logged for offline analysis. We have developed a component based software framework called THYME (Landsman and Alterman, 2003) that supports the rapid construction of groupware systems that include automatic logging and replay tools. The replay tools provide a VCR like console that allows all activity in each user's interface to be replayed for any recorded session.

To identify weak spots in the representational system, we use variety of discourse analysis techniques to analyze communication transcripts; details can be found in (Feinman and Alterman, 2003). We have developed a suite of analysis tools that work alongside the replay tools described above to assist in this analysis. These tools make it easy to tag communications traces, and perform a variety statistical operations and graphical analyses.

Once we have identified weak spots in the representational system, CRs are developed and added to the system. The fact that information captured by CRs is both highly structured and an integral part of the users' activity means that it is potentially very relevant to task-level intent inference, and, most significantly, can be leveraged by autonomous processes because of its structure. Common "off-the-shelf" AI techniques can potentially exploit this information to drive task-level adaptation. As with adaptation in direct manipulation systems, adaptive components may be developed that do not require the user to devote significant cognitive resources to using them, because the user does not have to explicitly communicate his or her intentions to the system.

We will refer to the revised representation system as a *practice-based language*; it is built by analyzing the practice that develops over time when a community of actors uses a groupware system. A practice-based language is employed in groupware systems both to improve user performance, and to provide sufficient information to support task-level intent inference (Figure 3b). A theoretical basis for practice-based languages can be found in (Alterman, 2000). Practice-based languages are not intent specific, but instead offer structure that help to make coordinating information easier for users exchange, organize, and interpret. The structure that is introduced serves to transform task-level information into a form that can be leveraged by an adaptive component.

In the remainder of this paper, we will examine how the development of a practice-based language for a groupware system called Vesselworld enables intent inference and the incorporation of an adaptive component. After introducing the Vesselworld platform, we will provide discourse examples to demonstrate the difficulty of performing intent inference without CRs. We will then describe one coordinating representation that helps users manage reference information, and present empirical data that shows that well designed CRs are adopted by users and improve their performance in the domain task. Next, we show how the information from the CR for reference management can be easily leveraged using a simple Bayesian Network to provide reliable task level intent inference. Finally, we present an adaptive

component that is driven by this intent inferencing system. Empirical data will be presented that demonstrates that the adaptive component further improves coordination and reduces cognitive load for the users.

# 4 Experimental Platform

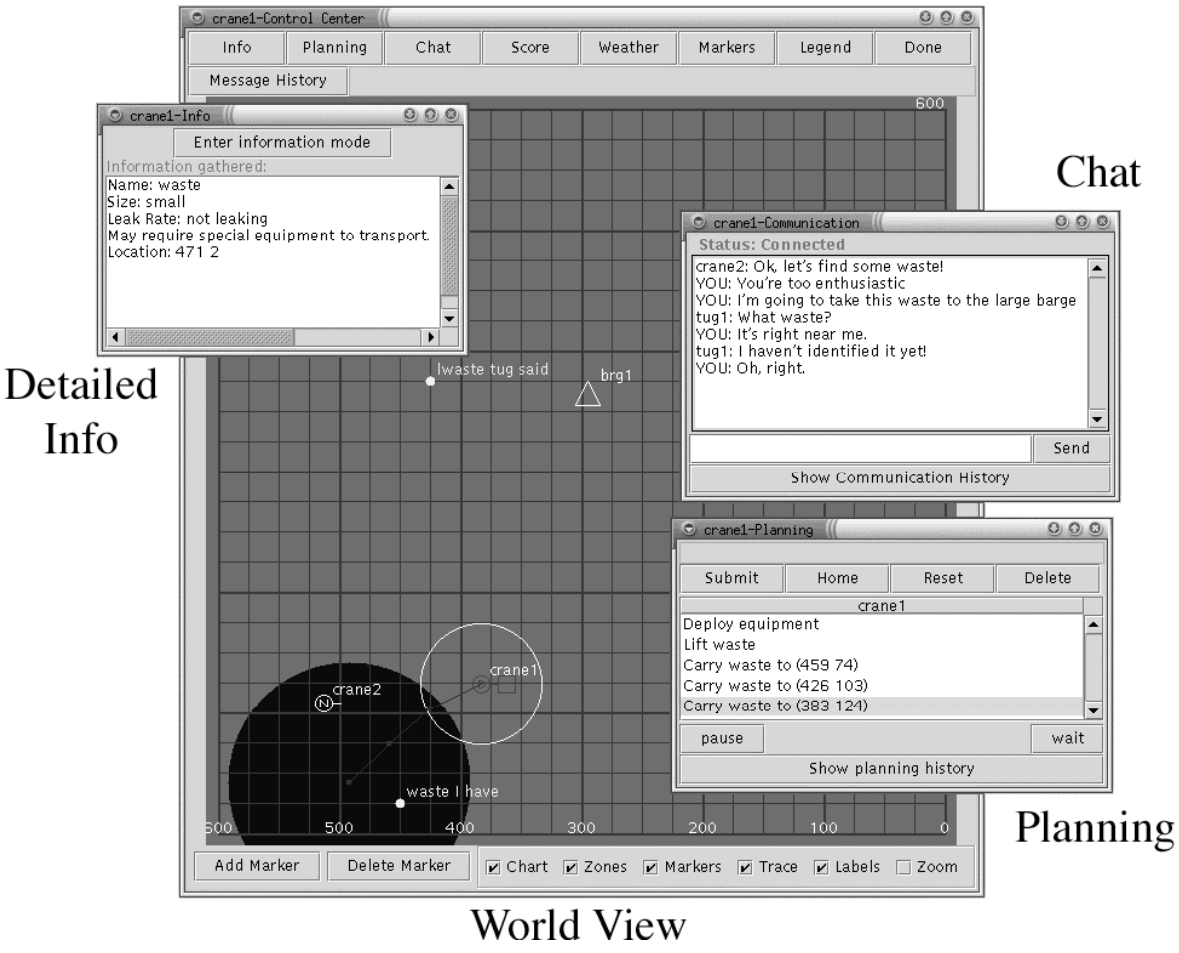



**Figure 4: Vesselworld Interface**

Our experimental platform is a groupware for planning system called Vesselworld (Figure 4). To support analysis, Vesselworld logs complete interaction data that can be used to "play back" user activity. This system was demonstrated at CSCW 2000 (Landsman, et al., 2001).

In Vesselworld, three participants collaborate on remote workstations to remove barrels of toxic waste from a harbor. Each participant is the captain of a ship, and their joint goal is to remove all of the barrels from the harbor without spilling any toxic waste. Two of the users operate cranes that can be used to lift toxic waste from a harbor and load them onto a large barge. The third user is the captain of a tugboat that can be used to drag small barges from one place to another. The crane operators can load multiple wastes on the small barge, and can unload them later.

The progression of a Vesselworld session is turn based, such that every user must submit a step to be executed by the server before the server can evaluate executions and update the world on each client screen. Users may plan any number of steps in advance, although any plan steps that involve objects are restricted to those objects that are currently visible, and only one step can be submitted to the server at a

time. Communication may occur at any point, but all communication must occur through a text-based chat window that is part of the system. The users are scored by a function that takes into account the time it takes to remove all of the waste, the number of barrels cleared, the number of errors made, and the difficulty of the problem.

Several complicating factors make careful coordination between participants a necessary part of solving a Vesselworld. Each ship has a geographically limited view of the harbor, thus ships in different locations will have different directly observable domain information, and no player has prior knowledge about the number and location of the wastes. The toxic waste barrels are of different sizes, which entail different coordination strategies that may involve more than one of the actors. For example, a single crane may lift a small or medium barrel, but the two cranes must join together to lift and carry a large barrel, and an extra large barrel may be jointly lifted but can only be carried on a small barge by the tugboat operator. Toxic waste barrels may require specialized equipment to be moved, and the cranes carry different types of equipment. Finally, the tugboat is the only actor who can determine the type of equipment a waste may require.

We performed a study (Clark, 2003) comparing the performance of several automated Vesselworld problem-solving strategies to human groups and found that even with human error, human groups significantly outperformed the automated strategies. The difference in performance was even more pronounced when only expert players were compared with autonomous strategies. The major finding of this study was that even though people encountered significant coordination errors and made other mistakes, they performed better than the automated solution because of their fluid use of coordination.

There are three versions of the Vesselworld system, corresponding to the phases of our methodology. The **base** version is the initial groupware system, and provides only a chat window for communication. The base version also provides each user with a private marker system that allows the user to place labeled markers on his or her private map. In Figure 4, markers are the small, labeled dots on the map. After an analysis of use of the base system, the **adapted** version was developed. The adapted system includes all of the functionality of the base system, and additionally three Coordinating Representations to support coordination. Finally, the **adaptive** version replaces one of the CRs (which was not used during user studies) with an adaptive component that is driven by information collected from another CR at runtime. The adaptive component offloads some of users' planning activity by predicting user's intentions and offering fully specified plans on the basis of these predictions.

## 4.1 Problems with inferring user intent from data in the base system

A major hurdle in developing adaptive systems is getting enough information about the user's activities without creating more work for the user or otherwise interfering with the user's domain task. Our solution to this problem is to introduce Coordinating Representations to the interface, and then leverage them to reduce the intent inference problem. Before describing how this approach was applied to the Vesselworld system, we provide several concrete examples that demonstrate why intent inference is difficult in the base version of the Vesselworld system.

To infer user intentions the system requires access to the current state of the problem, and to a library of plans that describe typical activity. In Vesselworld, state information consists of information about the positions and characteristics of toxic waste spills (size, type of equipment needed) for a given problem. In the base system, the only source for this information is online chat.

Figure 5 below is an excerpt taken from a chat transcript for a group of users using the base system. In the first four lines, tug announces information for four new wastes; note that in line one, the word

"small" at the end of the line is a typographic error, but none of the participants need to correct it. At line five, the tug operator announces the intention to move north. Lines six through twelve are concerned with planning the next move. In lines six and seven, both of the cranes independently suggest moving to 300, 350 to handle the extra large waste that was announced in line 1. In line 8, the tug suggests getting the large at 425, 525 instead of the extra large, but then retracts this suggestion in line 10 and lends support to the plan to get the extra large waste. In lines 11 and 12, the cranes confirm this plan.

1. tug1: extra large at 300 350 needs net small
2. tug1: small at 300 425 nothing
3. tug1: large at 425 525 nothing
4. tug1: medium at 350 450 dredge
5. tug1: will move north
6. crane1: Separate and move to 300, 350. deploy net and place on barge, then crane 2 can go get the small and crane 1 can get the medium.
7. crane2: separate and move near 300, 350
8. tug1: lets get the non extra large one first
9. crane1: move to near 425, 525 then?
10. tug1: forget my suggestion and go to extra large
11. crane1: OK.
12. crane2: ok

**Figure 5: Referring and planning in the base system**

At a minimum, to extract intent from the chat excerpt in Figure 5, it is necessary to identify and track references. For example, the reference in line 6 to "300, 350" has the same referent as the reference in line 1 – "xlarge at (300, 350)"; the reference in line 7 to "near 300, 350" also has the same referent as that in line 1. In the remainder of this section we will provide four specific examples of some of the features in Vesselworld chat that make accurate reference information very difficult to extract, both for the users, and autonomously.

### 4.1.1 Shifting Referential Conventions

| | |
|---|---|
| Tug1: extra large at 300 350 needs net | *Common referring expression used by one group. No shorthand was ever developed.* |
| Crane1: Legend: [Sm\|L\|XL] – [Ni [no id'd] Net \| Dr] | *Crane introduces a "legend." This formalism is used sporadically throughout the remainder of the dialog.* |
| Tug1: mN at 50 225 | *Common shorthand used in later problem solving sessions by one group – m = medium and N=net. This shorthand is never explicitly defined.* |

**Figure 6: Different ways of referring to wastes, between groups**

Referential conventions vary widely between groups. Some groups develop shorthand for referring expressions, and some groups do not. Some groups are explicit in defining this shorthand; other groups converge to common conventions gradually over time. For our purposes, the problem with such inconsistencies is that it is very difficult to produce a single rule or set of rules to identify referring expressions within chat. Figure 6 provides several examples of the various methods employed for speaking about objects; all examples were taken from different groups.

Even within groups, referential styles vary between participants and over time. Very often conventions will "compete," and as a result the group never converges to a single way of referring to objects. In Figure 7, a section of a transcript is shown for a group that exhibits such competing conventions. In the segment of dialogue shown, the players are announcing waste barrels they have found. In this group crane2 always includes an "@" between the type specifier and the location specifier of the referring expression, and a question mark is used to indicate that equipment requirements for the waste are unknown. Crane1 usually places the type description before the location, and tug places a type description after the location. The participants never converge to a common convention

1. Crane1 l 543 204
2. Crane1 s 562 150
3. Crane2 X?@150,559
4. Tug1 1 395 lg dredge
5. Crane2 s?@190,434
6. Crane2 s?@202,336
7. Crane1 sm 394 71
8. Crane1 large 395 22

**Figure 7: Different ways of referring to wastes, within one group**

### 4.1.2 Errors in recording waste information

Users make errors in recording waste information frequently, and these errors can persist throughout a significant portion of the dialogue, again making it difficult to identify accurate information about the world. An example of such a persistent error is shown in the two excerpts taken from a single problem session shown in Figure 8. In the first excerpt (the left pane), the tug mistakenly transcribes a small waste requiring no equipment (sX) at 275, 450 as being at 275, 250. In lines three and four, Crane2 requests a clarification and suggests a repair. Tug confirms the crane's correction, and notes the error in line 6. In the second transcript (the right pane), which takes place three minutes later, Tug again makes

the same error, at line 12, which results in Crane2 ultimately proceeding to the wrong location (line 18). In line 20, the tug finally catches the error and corrects the mistake (line 21).

| | |
|---|---|
| | *Three minutes later.* |
| 1. Tug1 sX at 275 250<br>2. Tug1<br>3. Crane2 ??<br>4. Crane2 275, 450?<br>5. Tug1 sX at 250 450<br>6. Tug1 er, yes<br>7. Crane2 ok | 8. Crane2: what was small at 275, 400?<br>9. Tug1: sX<br>10. Crane2: ahve sX at 450 above that<br>11. Tug1: mD at 400 350<br>12. Tug1: yes, there is an sX at 275 250 as well<br>13. Crane2: I have two amall at 275, 400 and 275, 450 are these the same?<br>14. Tug1: no, there are two sX there<br>15. Tug1: well, there are actually three in almost a line from n-s<br>16. Crane2: at 400 and 450? what about the 275, 250?<br>17. Crane2: ok, so the southern exists?<br>18. Crane2: I'm almost there, nothing at 275, 250<br>19. Tug1: 300 350, 250 400, and 275 250 are my markers<br>20. Tug1: argh<br>21. Tug1: I mean 275 450<br>22. Crane2: ok, those sound good<br>23. Tug1: i don't know why I kee doing that. |

**Figure 8: Two transcripts (three minutes apart) demonstrating confusion due to a transcription error**

### 4.1.3 Lack of precision

Users of the base Vesselworld system do not always specify wastes with precise locations when talking about them. Generally, it is not necessary for participants to have precise locations at the outset of a plan to handle a waste -- general directions are sufficient, thus, chat is peppered with relational (e.g. "next to") and directional (e.g. "east") references to domain objects. An example of this is shown in the excerpt in Figure 9. In the excerpt, the Tug operator introduces two toxic waste spills at line 1. The location of the first waste is approximate (there is no waste at 120, 415 in that problem), and second is relative to the first. This lack of precision does not cause problems for the other users (other users never ask for exact coordinates of either waste).

1. Tug1 There are two waste sites near me, one at 120/415, one just SE of it.
2. Crane1 what sixze/ equi[p?
3. Crane2 hmm shall we come up with a short hand for a waste info?
4. Tug1 The Se one is small, needs a Dredge, the first is small, and need NO equipment. Does that mean I can pick it up?

**Figure 9: Lack of precision in identifying wastes**

### 4.1.4 Inconsistent / Conflicting information

Because of the lack of precision in talking about wastes, users do not maintain the same information about where wastes are in the world. This leads to chat dialogues where different participants will refer to the same waste, but will use different locations in the referring expression. The resultant ambiguity renders resolution of references to actual domain objects impossible in these cases. It also confuses the participants themselves, and is a significant source of errors. In Figure 10, two excerpts from a single session are shown that illustrate this type of ambiguity. The two transcripts are ten minutes apart, and although four separate locations (lines 1, 5, 11, and 13) are mentioned, only two of these correspond to actual wastes. It is impossible to tell from these transcripts alone which of the locations are accurate (in this case, the actual wastes are at 105, 420 and at 130, 430).

1. Crane2: I found a waste at 120 420
2. Crane1: ok
3. Crane1: what type of waste?
4. Crane1: large,small?
5. Tug1: 105 420 needs a dredge, i think that is where you are
6. Tug1: small
7. Crane1: ok
8. Crane2: Thanks for checking
9. Tug1: no problem

*Ten minutes later.*

10. Tug1: 105 420 needs a dredge, other does not need anything
11. Crane2: dredge required on 120 412
12. Crane1: i'm there
13. Tug1: 130 430 none

**Figure 10: Two transcripts (10 minutes apart) demonstrating conflicting location information**

### 4.1.5 Implications for Plan Recognition

The above examples characterize some of the issues with chat that make the extraction of state information that can be used to infer users' intent very difficult. The examples also reflect some of the difficulties that users themselves face in maintaining a common view of the world. Lack of complete and correct information has several implications for user planning behavior:

- Planning is error prone, as it may be based on incorrect information. This is seen in Figure 8, when the crane proceeds to a location where there is no waste.
- Plans cannot always be complete, because complete information may not be available until execution time. Figure 9 shows how information about wastes can be exchanged without complete infor-

mation. Players may proceed to regions ("let's get the wastes in the NE corner") but do not complete their plans until the wastes are in fact visible.

- Planning in the base system is highly dependent upon local and recent information. Neither the users nor the system can easily discern faulty from accurate information. Plans are often only partially executed, and highly subject to change.

In summary, plan recognition is difficult in the base system not just because chat is hard to understand, but because activity itself is not as "planful" due to the users' difficulties in maintaining common ground.

## 4.2 Introduction of Coordinating Representations

The incorporation of Coordinating Representations in the adapted version of Vesselworld solves the problems described in the previous section. To develop CRs, we performed an analysis of use of the base version of the system. On the basis of our analysis, we developed three CRs. One of these CRs, called the Object List, was the primary source of state information in the adaptive system.

The Object List (Figure 11) is a WYSIWIS component that helps users to manage and coordinate referential information. It displays the same data in tabular format for each user. Users enter and maintain all of the data in the Object List. Each row of data in contains several fields of information, including a user assigned name, the status, and the location of the associated object. The location field may be filled in by clicking on the field and then on the object in the interface. The size, equipment, action, and leak fields are filled in using drop-down menus. A free text field ("notes") is also provided for each entry so that any other relevant information may be communicated. Entries in the Object List can be displayed on the Control Center (Figure 4) as icons that are annotated with the name that is in the "Name" field.

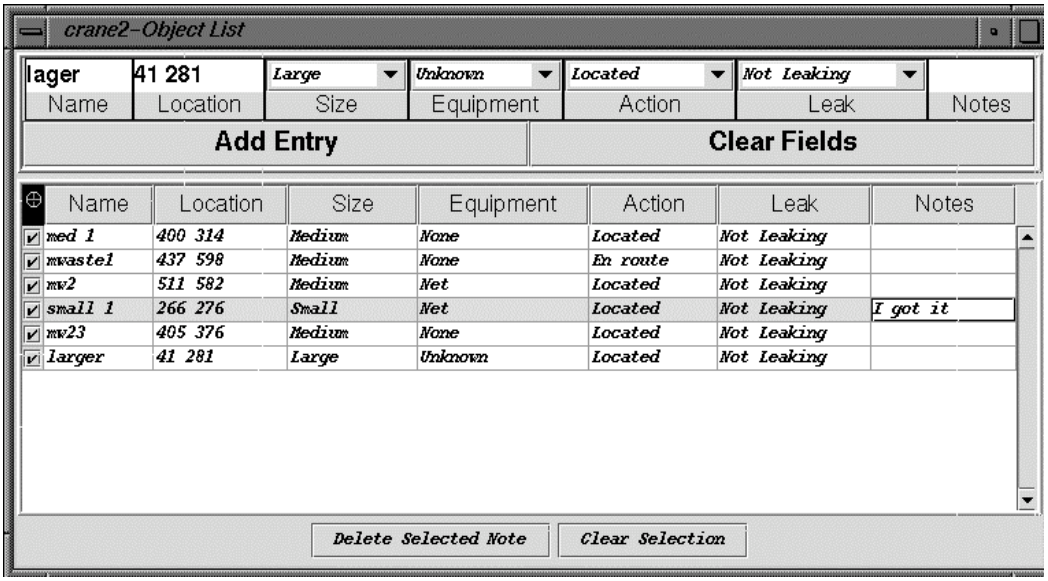



**Figure 11: The Object List**

As presented above, a CR should provide three features to support adaptation:

1. It introduces work that users want to do.
2. It will change how users organize their task in ways that improve their performance.

3. The information it captures can be leveraged by traditional AI techniques.

If the following section, we summarize a validation study (see Alterman, et. al., 2001a for details) that demonstrates that the Object List meets the first two of these criteria, and after that, we will provide evidence for the third.

### 4.2.1 Validation of the Object List

Crane1: I got an XL!

Tug1: I got nothing, you luck basrstartd.

Crane2: I got an Xl and an L, mommy! ;)

Tug1: Merry christmas kids….

Crane1: I'll map North third?

Tug1: I'll take middle 3rd.

Crane2: I'm at south-central. Tug, where are you?

Tug1: I'm jus nw of the barge, let me put that on the map…

Tug1: actually more w than n.

Crane2: With the LB in the corner, maybe our best bet is moving the SB NW and loading it with all the NW corner's goodies, which CRANE1: can map

Crane1: not a bad plan…

Tug1: Ok, I'll make a bit of a sweep around here while CRANE1: looks around.

Crane1: Tug, can you pick up the SB at your earlier opp?

Tug1: CRANE2: can map up on the way?

**Figure 12: Opening dialogue for Vesselworld with CRs**

Figure 12 shows the opening dialogue in a Vesselworld problem solving session where users had access to coordinating representations. This dialogue ensues before all of the participants have submitted their plan steps to the system for the first round of action. What is interesting about this dialogue is that nowhere is detailed information provided about any of the toxic waste barrels.

Figure 13 shows the Object List that is constructed during this dialogue. Only three of the entries into the Object List were mentioned in the opening dialogue, and none of these were explicitly named. Much of the dialogue that accompanied the discovery of a new waste in the groups using the basic system is now occurs offline outside of chat, mediated by the Object List. Identifiers are attached to each of the objects that are found; and these are used in future conversational exchanges about the wastes.

| ⊕ | Name | Location | Size | Equip | Action | Leak | Notes |
|---|---|---|---|---|---|---|---|
| ⊠ | G1 | 556 465 | XLarge | Unknown | Located | Not Leaking | |
| ⊠ | A1 | 186 107 | XLarge | Unknown | Located | Not Leaking | |
| ⊠ | m | 550 447 | Small | None | Located | Not Leaking | |
| ⊠ | A2 | 249 21 | Large | Unknown | Located | Not Leaking | |
| ⊠ | m | 250 149 | Small | None | Located | Not Leaking | |
| ⊠ | m | 449 349 | Small | None | Located | Not Leaking | |
| ⊠ | SB | 305 310 | XLarge | None | Located | Not Leaking | |

**Figure 13: The Object List constructed during the opening dialogue**

We performed a single-variable experiment to assess the impact of coordinating representations on the performance of teams of subjects using Vesselworld. One set of teams used the base Vesselworld system. The other set of teams used the adapted Vesselworld system, which incorporated CRs (including the Object List). Each set consisted of three teams of three subjects. The groups consisted of a mix of area professionals, mostly in computer-related industries, and undergraduate students; all were paid a flat fee for the experiment.

In general, the performance of groups with coordinating representations improved in many measures: clock time, number of system events generated (an indicator of interface work), and number of errors committed. Performance in the trouble areas we had identified in our analysis was notably improved, with errors due to miscommunication of object information significantly reduced.

Because of the relatively small sample population, individual differences, and the fact that user groups saw randomly chosen problems of varying difficulty, the variance of the raw data was quite high. To account for these problems, our results compare the final five hours of play for each group, and these results were normalized over a general measure of complexity for each of the problems solved. These results are shown in Figure 14.

| Indicator | Improvement (reduction) |
|---|---|
| **Communication** | 57% ($p<0.01$) |
| **Domain Errors** | 61% ($p<0.2$) |
| **System Events** | 38% ($p<0.06$) |
| **Clock time** | 49% ($p<0.01$) |

**Figure 14: Improvement of CR groups over non-CR groups; final 5 hours of play**

The most significant effect is the 57% reduction in communication generated. This confirms our observations that a significant amount of communication was offloaded into the CRs. Also highly significant is the 49% reduction in clock time. Only slightly less significant is the reduction in system events (mouse clicks, etc.), down 38%. These reductions were all expected. Also as expected, overall domain errors (errors in performing domain actions which led to a toxic spill) were reduced by 61%. The variance of this measure was quite high due to the low frequency of errors; this reduced its confidence below statistical significance ($p<0.2$).

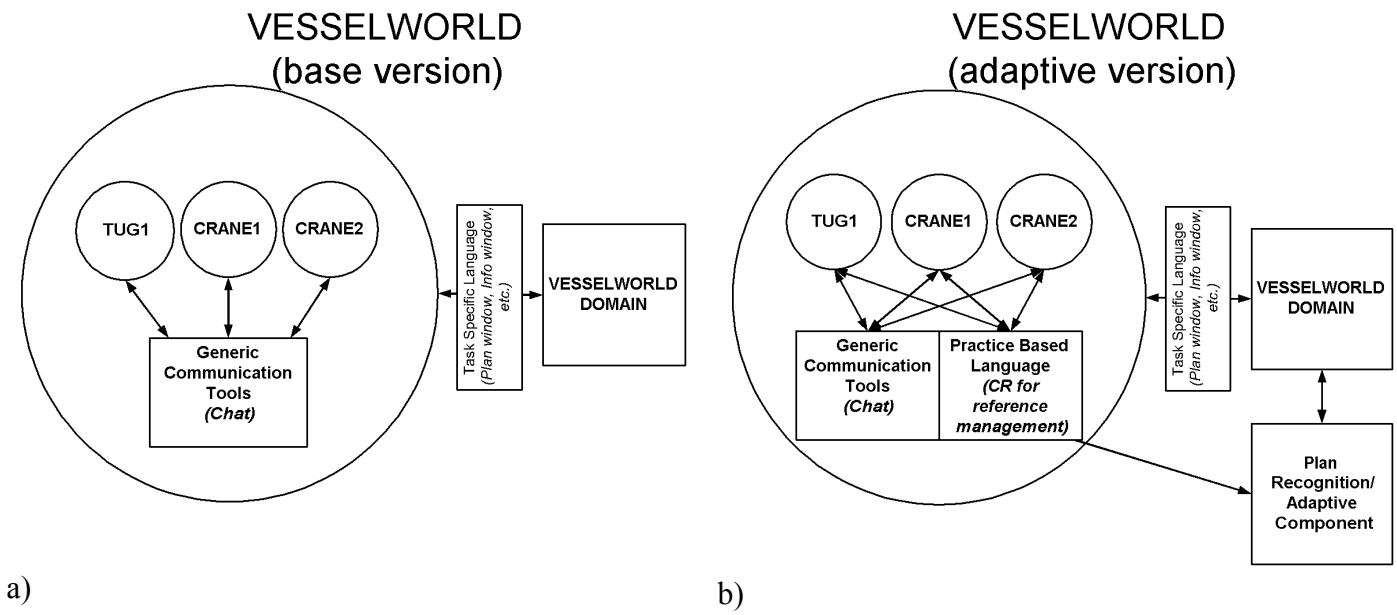
## 4.3 Summary

The results above demonstrate that, when CRs are designed correctly, collaborators prefer to use them instead of generic communications tools (like free text chat) for the exchange of certain kinds of coordinating information, and CRs will improve users performance in the task domain. In Vesselworld, the Object List was used to support coordination over domain objects. Vesselworld participants universally adopted the Object List, and it formed an integral part of their practice. During a typical Vesselworld session with the adapted system, every waste found by the participants is added to the Object List. Subsequent discussions about wastes in chat use the names entered into the Object List, and the information in the Object List was maintained to reflect the current state of the world.

The Object List is part of a practice-based language for Vesselworld, via which users indirectly provide information to the system that can be harnessed by a plan recognition system. Use of the Object List is neither burdensome to the users nor peripheral to the domain task itself, as in the case of intent expres-

sion languages like ADL (used in Collagen). Instead, the Object List improves the users' task; they want to use it, and its use supports more complete plans and less error-prone domain activities.

# 5 Inferring user intent with CRs




a) b)

**Figure 15: Application of our methodology to Vesselworld. a) The base version of Vesselworld; b) The adaptive version, which uses information from a CR for reference management**

The Object List produces structured and unambiguous information about the state of the world at run-time. In this form, state information is immediately available for further back-end processing; it does not need to be parsed, or otherwise interpreted.

As shown in Figure 15a, the base version of Vesselworld provides only a chat tool so that people may share coordinating information. We have shown how it is difficult to extract reference information from chat, and how this makes plan recognition very hard. We now describe how we are able to use the reference information captured by the Object List CR for reference management to produce reliable, task-level intent inference, which in turn forms the basis for our adaptive component (Figure 15b).

To perform intent inference, we use a Bayesian Belief Network (Pearl, 1988) that generates a likelihood estimate for every possible agent-waste-operator tuple at each update to the available evidence, where wastes are taken directly from the Object List, and operators come from a set of task-level goals we have defined. We are only concerned with relative likelihood estimates, and not actual probabilities of each goal. We used an off-the-shelf software package called Netica to build our network (Norsys Software Corp., www.norsys.com).

Figure 16 portrays a simplification (two interior nodes and nodes for the tug operator have been omitted to simplify the diagram) of the Belief Network that is currently being used in Vesselworld. Data is accumulated from the Object List, chat transcript, and plans, and posted to the unshaded nodes in the figure. The interior nodes (shaded) are incorporated primarily to reduce the total number of conditional probabilities entries required by the model. The two output nodes in the diagram are two-state nodes (yes or no), reflecting the likelihood that an agent has the intention represented by that node. The structure of the network reflects the view that user intention (to load or lift a waste) is caused by some set of

observable conditions (the location, heading of a user, equipment needs for a waste, etc.). Obviously, this is an over-simplification of how intention might actually arise, but it suffices for our purposes.

Nodes are classified into three broad categories to serve as general guidelines for building BNs in other domains that employ Coordinating Representations:

- The "Coordination Information" nodes reflect a judgment as to how likely a reference to a waste (taken directly from the Object List) that has appeared recently in chat is an indicator that a plan will be executed. As co-referencing activities are pervasive in collaboration (Clark and Brennan, 1991; Clark and Wilkes-Gibbs, 1990), we expect that other collaborative domains would benefit from CRs that provide similar referential information.
- The "State Information" nodes compute the support for each possible plan given state information. This includes information such as whether the type of equipment is appropriate, the size of the waste (which determines how many actors must be involved with the waste), and whether the agent is holding something. Some of this information (equipment and size) is derived from the directly from the Object List. The other information ("Barge Full" and "Agent Holding") is derived from prior plan execution activities.
- The "Heuristic Information" nodes are heading and proximity, which may be derived from the positional information of objects in the Object List, and planning information from the users.

The output of the network is a likelihood that an intention to achieve some goal exists. The output is passed back to the system to be used as described in the next section.

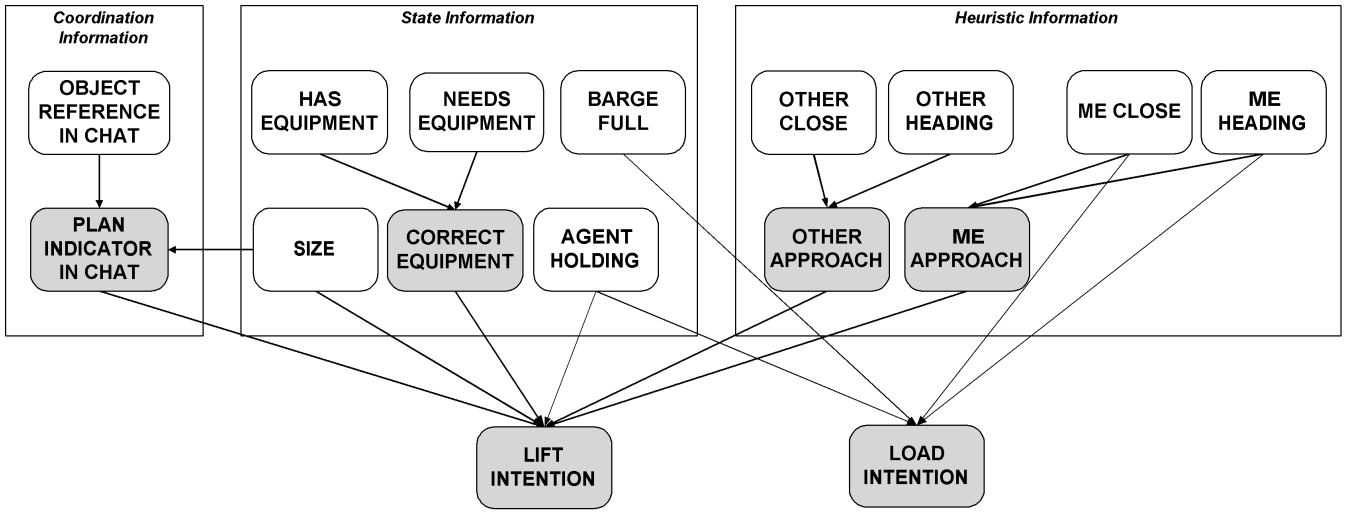



**Figure 16: The Vesselworld Belief Network**

The Belief Network produces new assessments each time the state of the world changes due to a plan submission, a new reference is detected in chat, or when the Object List is modified. When this occurs, new evidence is posted to the network for each possible pairing between objects in the Object List (that have positional information) and actors, and a likelihood is obtained for each agent-operator-object tuple. The tuple with the highest output likelihood greater than some cutoff (empirically determined) is the most likely intention. In preliminary validation with our network against collected datasets, we found that nearly all of the goals were correctly predicted, and that the false positive rate was about 25%.

## 5.1 An Adaptive Component

To avoid interrupting the user's task and creating more work for the user, we drew some insights from McFarlane's (1997) taxonomy of interruption in the design of the interface to the adaptive component. This work suggests that even though some information is ignored when a user is able to choose when and how much they attend to interrupting information, this information is retained better and the user's main task performance suffers less than with other methods of interruption. Thus, we provide information to the user in a passive interface component that leaves the decision if, and when, to use the provided information up to the user.

The interface to the adaptive component is shown in Figure 17. The tool provides a means for people to construct individual as well as shared plans (Grosz and Sidner, 1986; Grosz and Kraus, 1996), and automates some of the mundane work involved in this process. In particular, once the system knows that more than one user shares a specific goal, it can generate a fully instantiated shared plan that will succeed if the system's information about the world is correct (and the users submit the plan to completion).


**Figure 17: The adaptive component**

The automatic generation of plans is possible using information from the Object List, and does not require the intent inferencing process described above. However, at any point in time over a hundred goals may be possible for each user, depending on the number and types of objects in the world. Intent inferencing is able to drastically reduce the set of possible goals to a small number of plausible goals that a user can select from.

The function of the adaptive component is as follows:

1. At each update to state information, (e.g. plan execution, information added to the Object List, a reference to an object mentioned in chat, etc.) the system offering each user up to five possible

goals (corresponding to the most highly ranked tuples output by the Belief Network), displayed top down and color-coded according to the systems "confidence" in that prediction. A user can only select a goal from their column.

2. When a goal is confirmed, it is copied into the top row, which displays all users currently confirmed goals. The user then has the option to retrieve an automatically generated plan that will accomplish the goal (the "Get Plan" button in Figure 17).
3. The system generates a plan that the user can inspect. In cases where the goal involves multiple actors, the other actors are invited to join the plan. If all invited actors do not accept the invitation, a plan is not generated, and the user is informed of the problem.
4. If the user accepts the plan (by clicking the "Accept" button in Figure 17), it is automatically copied into the users planning window for execution.

If the plan is generated from correct state information, and no user modifies the state in such a way that conflicts with the generated plan, the plan will succeed.

## 5.2 User Studies

To evaluate the effectiveness of the adaptive component, we performed a 40-hour study with four teams of three people. The players were a mix of students and local-area professionals, with varying degrees of computer proficiency. Each team was trained together for two hours in use of the system, and then solved randomly chosen Vesselworld problems for approximately ten hours. To alleviate fatigue concerns, the experiment was split into four three-hour sessions.

The participants were divided into two populations of two teams each. One population was tested on the adapted system, which contained all three CRs but did not have the adaptive component. The other population was tested on the adaptive system, in which the adaptive component replaced one of the CRs.

Our analysis sought to answer the following questions:

- Was the adaptive component used? An adaptation can only be useful if people are willing to use it.
- Did it improve performance? Even if an adaptation is used, it may or may not improve user performance.
- Did it reduce user effort? Even if an adaptive component is used, and improves performance, it does not necessarily reduce user effort.

In the following, data is presented for the last 5 hours of play time for each group, by which time performance with the system had stabilized. We show that the above questions can be answered affirmatively.

### 5.2.1 Was the adaptive component used?

| | Confirmation Frequency | Requests Per Conf | %Requests Accepted | %Accepted Completed |
|---|---|---|---|---|
| Avg. | 0:01:32 | 1.31 | 71.02% | 82.88% |
| Std. Dev | 0:00:46 | 0.30 | 18.88% | 6.74% |

**Table 1: Usage statistics for the adaptive component**

All groups used the adaptive component to generate plans within the system. Table 1 shows usage statistics for the adaptive component as rate of activity. Our evidence show that system predictions were confirmed, and furthermore, plans were frequently requested, accepted and executed.

The first column in Table 1 shows that a confirmation of a goal (when a user clicked on a prediction) was made by a user roughly every minute and a half during a problem solving session (sessions last anywhere from 45 min. to two hours). The second column shows that there were at least as many requests for plans as confirmations of goals (though not necessarily a plan request for each confirmation), and that multiple requests were occasionally made for a confirmation. The third column shows that roughly 71% of the plans that were requested were accepted (this number reflects the proportion of requested plans that were accepted, and not the number of confirmed goals that resulted in an accepted plan), and the final column shows that 83% of the plans accepted were actually executed to completion (every step was submitted to the server).

| | Steps | Crane Goals |
|---|---|---|
| Avg. | 24.60% | 43.34% |
| Std. Dev. | 7.59% | 14.69% |

**Table 2: Proportion of steps accounted for by the adaptive component**

In Table 2, we examine the proportion of steps that were submitted to the server that were generated by the adaptive component. The first column (Steps) shows that nearly a quarter of all plan steps submitted to the server (where an average game contains about 380 total submitted individual steps) were generated by the adaptive component. The number in the second column shows the percentage of goals accomplished by the cranes that were the result of the adaptive component. Only goals that could have been the result of plans produced by the adaptive component are reflected in this number.

### 5.2.2 Did the adaptive component reduce cognitive effort?

To quantify the impact of the adaptive component on the amount of effort required during planning, we examined the mean duration between steps submitted to the server for plans that were generated manually, versus those that were generated automatically. These results are summarized in Table 3.

| Automatically generated plans vs. manually generated | Percentage reduction in mean plan step duration | Significance |
|---|---|---|
| Non-adaptive groups | 52% | p<.01 |
| Adaptive groups, manual plans | 57% | p<.01 |

**Table 3: Reduction in plan step duration time for automatically generated plans**

The amount of time taken by users between submitting subsequent steps of automatically generated plans was significantly less than that for both the groups that did not have the adaptive component, and for manual planning phases of activity for groups that did have the adaptive component. This corroborates the error data above, which indicates that coordination was less difficult for automatically generated plans. We conclude from this result that the adaptive component reduced the workload of the collaborators.

### 5.2.3 Did the adaptive component reduce errors?

Joint errors account for a significant proportion of total errors made in Vesselworld (Alterman, et. al. 2001a), because they require close coordination, and thus the heightened attentions of the participants.

The groups that used the adaptive component had 45% (p=.069) fewer joint errors per minute than the groups that did not. This reduction demonstrates a strong tendency.

The reduction in joint errors confirms prior analysis of use of the Vesselworld system. The adaptive component produces coordinated plans in advance, and users may simply submit each step and be assured that actions will be coordinated.

### 5.3 Summary

Our validation study demonstrates that the adaptive component provided useful and usable adaptive support to users of the Vesselworld system. The adaptive component was heavily used, reduced coordination errors, and reduced cognitive load. These results demonstrate how a practice-based language, based upon the analysis of coordination work in a groupware system, can be leveraged to produce useful adaptations that do not introduce more work for the user. These results provide support for our overall methodology.

## 6 Discussion

As we have previously discussed, the design of an interface language is a fundamental consideration when trying to introduce adaptive capabilities to an existing system. When examining an interface language with respect to its ability to serve as a front end to an adaptive system, it is useful to consider the following three questions:

1. What information will be provided to the adaptive component?
2. Is the user willing to do the work required to provide this information?
3. How hard is it to use this information once it is obtained?

The utility of any adaptive component directly depends on how each of these questions are answered; for some kinds of information, it may be harder to find an interface language that is both usable, and provides information that is easy for an adaptive component to leverage. In this paper, we have described three classes of interface languages, which answer these questions in different ways. These are summarized here.

Within the class of *task-specific interface languages*, some kinds of direct manipulation systems are able to provide the right kind of information to support adaptation. The important features of these systems are: the operators available in the interface are very similar to the users task-level goals, and; the system and user share a common representation of data relevant to the task. For such systems, the above questions may be answered as:

1. Two kinds of information are provided to the adaptive component; task-level goal information, and complete information about the state of the domain.
2. The user is willing to use the interface; this is the sole interface through which the user interacts with the domain, and various studies have documented people's positive experiences with such systems.
3. It is easy to use this information to support adaptive components because it is precisely the information required to infer users' task-level intentions.

As discussed above, such systems are not possible in all domains. For these domains, *intent expression languages* are employed to solicit a user's intentions directly, regardless of whether the intent expression

language is formal or natural. We have discussed a range of systems that use such languages. For these systems, we answer the above questions as:

1. The information provided to the system is information about user intent. Systems that employ intent expression languages attempt to solicit users' intentions directly.
2. Whether or not people are willing to use intent expression languages depends on the design of the language. People are not generally happy using formal, highly constrained intent languages; it is often easier for a user just to *do* something than to explain it to a system with a structured language (Lesh, et al. 1999). People may be willing to use natural language, but current limitations of the back-end NLP algorithms can lead to user frustration (Schneiderman, 1998).
3. The ease with which information collected via intent expression languages again depends on the type of language used. Information that is provided via a formal, highly constrained language is easily leveraged. Information provided via natural language or artificial languages that do not constrain conversational interaction is very hard to leverage using current AI techniques.

We have introduced practice-based languages, which can be applied to groupware systems. Our approach focuses upon re-engineering the representational system by introducing Coordinating Representations. Our answers to the above questions are:

1. CRs provide the system with access to coordination information that is difficult for people to maintain and/or exchange, yet is crucial to successful coordination.
2. People willingly use CRs because CRs make coordination easier. We have collected empirical data that demonstrates this.
3. Information can be easily leveraged for adaptation because it is highly structured. We have demonstrated how this may be done using off-the-shelf AI technology, and have presented empirical evidence that validates the effectiveness of the resultant adaptation.

CRs are developed where there are weak spots in the representation system; we identify a need for some external representation in an existing (groupware) system. As has been observed in other studies, CRs become integral parts of a joint practice, and people communicate about the domain through CRs and in reference to them (e.g. Suchman and Trigg, 1991; Hutchins, 1995a). People employ CRs to manage information about the features of the state of the world that are relevant to coordination, in a compact and structured form.

The information that passes through a CR is tuned to domain task, and is maintained by people so that it is up to date. It can be easily leveraged by traditional AI techniques to introduce adaptations that are responsive to the users' joint task. In this paper, we have shown how a belief network can be used to perform intent inference using reference information from a CR. We used a BN because it was easy and powerful enough to do what we needed. Other AI techniques might be used; the important point is that it is because of the CRs that it is easy to apply "off-the-shelf" AI technology.

The methodology we have described is concerned with collaborative applications. We have described a way to develop CRs by analyzing available interaction data. In synchronous groupware environments, interaction data can often be easily collected, because most or all of the users' interactions are mediated by the system. It is also possible to develop CRs for asynchronous groupware using similar analytical methods.

# 7 Conclusions

We have described a methodology for adding adaptive functionality to groupware systems. Our approach focuses upon re-engineering the representational system so that it: 1) better supports users' ability to manage coordinating information; and, 2) captures information in a form that can be leveraged by traditional AI procedures. We have provided technical detail about how this information can then be leveraged using a Bayesian Network to infer user intentions, and have presented evidence that demonstrates that our approach accurately infers user intentions, and can be used to drive an adaptive component that further improves users' ability to coordinate.

Generally, our approach differs from more common approaches that focus on developing the back-end AI technology that will drive the adaptive system. Regardless of the power of the technique, an adaptive component is ultimately limited by the amount and type of information it can get from the user about the current state of the domain task. For development efforts that seek to implement real-world adaptive systems, it is equally or potentially more important to consider the design of the interface language.

We have described an approach to re-engineering the interface language by adding coordinating representations that fix weak spots in the existing representational system, and have shown how the addition of CRs supports the introduction of powerful adaptations using standard techniques. The approach is applicable to all collaborative domains, and is a step in providing well-integrated computational support tools that are sensitive to context, and can respond intelligently to the needs of collaborators.

# Acknowledgments

This research was supported by the Office of Naval Research under grants No. N00014-96-1-0440 and N66001-00-1-8965.

# References

1. Allen, J. and Ferguson,G. (2002). Human-Machine Collaborative Planning. To appear in Proceedings of the Third International NASA Workshop on Planning and Scheduling for Space, Houston, TX, October 27-29.
2. Alterman, R., (2000). Rethinking Autonomy. Minds and Machines, 10:1 15-30.
3. Alterman, R., Feinman, A., Landsman, S., Introne, J. (2001a). Coordination of Talk: Coordination of Action. Technical Report CS-01-217, Department of Computer Science, Brandeis University.
4. Alterman, R., Feinman, A., Landsman, S., and Introne, J. (2001b) Coordinating Representations in Computer-Mediated Joint Activities. Proceedings of the 23rd Annual Conference of the Cognitive Science Society, 43-48, 2001.
5. Bödker, S. (1991). Through the interface: A human activity approach to user interface design. Hillsdale, NJ: Lawrence Erlbaum Associates.
6. Clark, C. (2003) A tightly coupled algorithm for playing VesselWorld. Technical Report CS-04-253, Department of Computer Science, Brandeis University.
7. Clark, H., and Brennan, S. (1991) Grounding in communication. In J. Levine, L.B. Resnik, and S.D. Teasley, editors, Perspectives on Socially Shared Cognition, pages 127-149.
8. Clark, H. and Wilkes-Gibbs, D. (1990) Referring as a collaborative process. Cognition, 22:1-39.

9. Cox, M. T. (2003). Planning as mixed-initiative goal manipulation. In the Proceedings of the Workshop on Mixed-Initiative Intelligent Systems at the 18th International Conference on Artificial Intelligence. Menlo Park, CA: AAAI Press.

10. Cox, M. and Veloso, M. (1997a). Supporting Combined Human and Machine Planning: An Interface for Planning by Analogical Reasoning. In Leake, D. & Plaza, E. (Eds.) Case-Based Reasoning Research and Development: Second International Conference on Case-Based Reasoning, pp.531-540. Berlin. Springer-Verlag.

11. Cox, M. and Veloso, M. (1997b). Controlling for Unexpected Goals when Planning in a Mixed-Initiative Setting. In E. Costa & A. Cardoso (Eds.) Progress in Artificial Intelligence: Eighth Portugese Conference on Artificial Intelligence (pp. 309-318): Berlin. Springer.

12. Cypher, A (1991): Eager: Programming Repetitive Tasks by Example. In: CHI '91 Conference Proceedings. (Eds: Robertson, Scott P; Olson, Gary M; Olson, Judith S) ACM Press, New York, 33-39.

13. Ellis, C.A., Gibbs, S.J., and Rein, G.L. (1991) Groupware: some issues and experiences. Communications of the ACM (34), pages 38-58.

14. Feinman, A., and Alterman, R. (2003) Discourse Analysis Techniques for Modeling Group Interaction. In Brusilovsky, P., Corbett, A., and de Rosis, F, editors, Proceedings of the Ninth International Conference on User Modeling, pages 228-237.

15. Ferguson, G. and Allen, J. (1998). TRIPS: An Intelligent Integrated Problem-Solving Assistant, in Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98), Madison, WI. pp. 567-573.

16. Goodwin, C. and Goodwin, M. (1996) Formulating Planes: Seeing as a situated activity. Cognition and Communication at Work. 1996. N.Y.: Cambridge University Press.

17. Grosz, B. and Kraus, S. (1996) Collaborative Plans for Complex Group Action. In Artificial Intelligence. 86(2), pp. 269-357

18. Grosz, B. and Sidner, C. (1986) Attention, Intentions, and the Structure of Discourse. Computational Linguistics, 12:3

19. Hutchins, E. (1995a) Cognition in the Wild. MIT Press.

20. Hutchins, E. (1995b) How a cockpit remembers its speeds. Cognitive Science 19, pages 265-288.

21. Hutchins, E.L.; Hollan, J.D.; Norman, D.A.(1986) Direct manipulation interfaces. In D.A. Norman & S.W. Draper (Eds.) User Centered System Design, New Perspectives on Human-Computer Interaction. Lawrence Erlbaum Associates, Inc.

22. Landsman, S., Alterman, R., Feinman, A., Introne, J. Vesslworld and ADAPTIVE (2001), Brandeis University Tech Report CS-01-213; Presented as a demonstration at Computer Soupported Cooperative Work, 2000.

23. Landsman, S., and Alterman, R. (2003) Building Groupware On THYME. Tech Report CS-03-234, Department of Computer Science, Brandeis University.

24. Lesh, N, Rich, C., and Sidner, C.L. (1999). Using plan recognition in human-computer collaboration. In Proc. 7th Int. Conf. on User Modeling, pp. 23—32.

25.McFarlane, D. C. (1997). Interruption of people in human-computer interaction: A general unifying definition of human interruption and taxonomy. (NRL Formal Report NRL/FR/5510–97–9870): Naval Research Laboratory, Washington, DC.

26.McTear, M. (2002), Spoken dialogue technology: enabling the conversational user interface, ACM Computing Surveys, vol. 34, pp. 90 - 169.

27.Miller, R., and Myers, B. (2002) Multiple Selections in Smart Text Editing. In Proceedings of Intelligent User Interfaces, San Francisco, CA., 103-110.

28.Nardi, B. (1993). A Small Matter of Programming. The MIT Press. Cambridge, MA. 1993.

Norman, D. A. (1991). Cognitive artifacts. In J. M. Carroll, editor, Designing interaction: psychology at the human-computer interface, pages 17--38. Cambridge University Press.

29.Oppermann, Reinhard (Ed.) (1994): Adaptive User Support. Hillsdale: Lawrence Erlbaum Associates.

30.Pearl, J. (1988). Probabilistic Reasoning in Intelligent Systems. San Francisco, Calif.: Morgan Kaufmann.

31.Rich., C. and Sidner, C. L. (1998). COLLAGEN: A Collaboration Manager for Software Interface Agents. User Modeling and User-Adapted Interaction. 8(3-4):315-250.

32.Sidner, C.L (1994). An Artificial Discourse Language for Collaborative Negotiation. Proceedings of the Twelfth National Conference on Artificial Intelligence. AAAI Press. Menlo Park, CA. pp. 814-819.

33.St. Amant, R and Cohen, P. (1998). Intelligent support for exploratory data analysis. Journal of Computational and Graphical Statistics, (4):545-558.

34.St. Amant, R., Young, R. (2001). Interface Agents in Model World Environments. AI Magazine 22(4): 95-108.

35.Shneiderman, B. (1983). Direct manipulation: a step beyond programming languages. IEEE. Computer. 16:57-69.

36.Schneiderman B., (1998) Designing the User Interface, Addison Wesley, 1998.

37.Suchman, L. and Trigg, R (1991). Understanding Practice: Video as a Medium for Reflection and Design. In Joan Greenbaum & Morten Kyng (eds.), Design at Work: Cooperative Design of Computer Systems, Lawrence Erlbaum Associates, Hillsdale, New Jersey, February.

38.Terveen, L. (1994) An Overview of Human-Computer Collaboration. Knowledge-Based Systems, 8 (2-3), 1994.

39.Terveen, L., Stolze, M., Hill, W. (1995) From 'Model World' to 'Magic World'. In *SIGCHI Bulletin*, 27 (4), 31-34.

40.Walker, M., Fromer, J., Fabbrizio, G.D., Mestel, C. & Hindle, D (1998) What can I say? Evaluating a spoken language interface to email. In Proceedings of the Conference on Computer Human Interaction (CHI 98), pp.582-589.