User Model User-Adap Inter (2006) 16:249–280
DOI 10.1007/s11257-006-9009-2

ORIGINAL PAPER

Using shared representations to improve coordination and intent inference

Joshua Introne · Richard Alterman

Received: 10 October 2005 / Accepted in revised form: 4 April 2006 /
Published online: 26 September 2006
© Springer Science+Business Media B.V. 2006

Abstract In groupware, users must communicate about their intentions and maintain common knowledge via communication channels that are explicitly designed into the system. Depending upon the task, generic communication tools like chat or a shared whiteboard may not be sufficient to support effective coordination. We have previously reported on a methodology that helps the designer develop task specific communication tools, called coordinating representations, for groupware systems. Coordinating representations lend structure and persistence to coordinating information. We have shown that coordinating representations are readily adopted by a user population, reduce coordination errors, and improve performance in a domain task. As we show in this article, coordinating representations present a unique opportunity to acquire user information in collaborative, user-adapted systems. Because coordinating representations support the exchange of coordinating information, they offer a window onto task and coordination-specific knowledge that is shared by users. Because they add structure to communication, the information that passes through them can be easily exploited by adaptive technology. This approach provides a simple technique for acquiring user knowledge in collaborative, user-adapted systems. We document our application of this approach to an existing groupware system. Several empirical results are provided. First, we show how information that is made available by a coordinating representation can be used to infer user intentions. We also show how this information can be used to mine free text chat for intent information, and show that this information further enhances intent inference. Empirical data shows that an automatic plan generation component, which is driven by information from a coordinating representation, reduces coordination errors and cognitive effort for its users. Finally, our methodology is summarized,

J. Introne (✉) · R. Alterman
Department of Computer Science, Brandeis University, Volen Center for Complex Systems,
415 South Street, Waltham, MA 02454, USA
e-mail: jintrone@cs.brandeis.edu

R. Alterman
e-mail: alterman@cs.brandeis.edu

and we present a framework for comparing our approach to other strategies for user knowledge acquisition in adaptive systems.

Keywords Groupware · Knowledge acquisition · Adaptive user interfaces · Coordinating representations · Plan recognition

1 Introduction

Acquiring enough run-time information about the user to provide intelligent support is a central issue in building any adaptive system (Fischer 2001; Jameson 2002; St. Amant and Young 2001). To tutor a student, a system needs to know the student's strengths and weaknesses; to offer context sensitive help, it needs to know what the user wants to accomplish; to help find relevant content, it needs to know the user's interests. For a collaborative system to adapt to its users' needs, it must have information that is relevant to collaboration—information like the users' shared goals, common knowledge, or roles.

A major hurdle in the design of a user knowledge acquisition strategy is how to encourage the user to furnish the information that the system needs without impairing the overall usability of the system. There are numerous approaches to the problem. Mixed initiative systems (e.g., Maes 1994; Horvitz 1999) often obtain information through agents that engage the user to jointly solve a domain task (e.g., TRAINS, Ferguson et al. 1996; LUMIERE, Horvitz et al. 1998; COLLAGEN, Rich and Sidner 1998). Intelligent tutorial systems exploit the information gained during the testing process to derive models of students (e.g., Anderson et al. 1995). Multi-modal systems (e.g., Bosma and Andre 2004) and affective interfaces (e.g., Kaiser 2005) integrate other sources of information, like gestures and biometrics, to enhance the information available to a back-end inference process.

In groupware systems, much of the information required to drive intelligent support is already available in the communication that goes on between coordinating users (see Alterman 2000). Unfortunately, most user communication occurs via unstructured channels such as chat or message boards, making it difficult for the system to extract the necessary information. One solution is to enlist a human actor as interpreter for the system; for example, in the COPPER system (Read et al. 2006) a student tutor evaluates a group process and provides this information in a form the system can use. The student tutor performs the group evaluation as part of her role and the system provides support for the activity.

However, it is not always possible to co-opt an individual's existing domain task to furnish the adaptive component with the structured information it needs at runtime. In these cases it may be possible is to add more highly structured communication channels to the system (e.g., Malone et al. 2001; Soller 2004). The structured information generated by collaborators when using these channels can be leveraged to populate user models and adapt the system. For instance, McLaren et al. (2006) show how students' use of a domain specific structure (Petri-nets or UML diagrams) can be used to infer metrics of group effectiveness. Similarly, Suebnukarn and Haddawy (2006) show how the collaborative use of a graph based interface tool can be mapped directly into the system's Bayesian Network based model of each student's conceptual knowledge. In both of these systems, interface structure is leveraged to simplify a back-end adaptive process.

Identifying the right kind of structure to introduce at the interface is a hard design problem. Care must be taken not to impair the users' ability to communicate with one another or perform their domain tasks, and there is no guarantee that the structure which is introduced will provide the system with the runtime information it needs. In this article, we describe a method for introducing structure that people will use, that helps them stay coordinated, and at the same time provides useful information to the adaptive system. The method hinges upon the development of task specific communication tools that support the exchange of structured information relevant to coordination. Because these artifacts add structure to communication, the information that passes through them can be easily exploited by runtime algorithms. Because the information is relevant to coordination in the task domain, it will be useful for domain-specific intent inference. Here, we present a series of case studies demonstrating the approach in a testbed groupware environment.

1.1 Coordinating representations

There is a rich body of ethnographic research that describes how people design and use structured artifacts to support specific and complex coordination as part of an established work practice (Suchman and Trigg 1991; Goodwin and Goodwin 1996; Hutchins 1995; Schmidt and Simone 1996). These kinds of artifacts have several common properties. They are external representations that address recurring problems of coordination within a community. They make it easier for individuals to align their private views of the world. There is a protocol that describes their typical use within a particular community of practice. Following Suchman and Trigg (1991), we refer to these kinds of artifacts as *coordinating representations* (CRs) (Alterman et al. 2001).

There are many examples of CRs in everyday life. The stop sign is a CR that helps people to coordinate their behaviors at an intersection. The arrivals and departures board at a train station supports coordination between many people and the operators of a railway terminal. The grocery list on the wall in the kitchen helps a couple coordinate the activity of supplying a household. The stock market "ticker" helps millions of people coordinate their market activities.

There are also many examples of CRs that have evolved over time to support specialized communities. For instance, the *complex sheet* in an airport setting is an example of a paper-based coordinating representation that helps coordinate transfers of baggage and people between gates in an airport during "complexes" (Suchman and Trigg 1991). Complexes are pre-scheduled periods of time during which all gates fill with incoming planes, transfers are made, and then all planes depart. As shown in Fig. 1, the complex sheet is a matrix mapping incoming to outgoing planes, and cells in the matrix are used to indicate the transfer of people or baggage. Planes are ordered chronologically along both axes, so completed transfers are checked off diagonally downward and to the right across the cells of the matrix.

As computers have become common in work settings, computer-based coordinating representations have been developed. The bug-report form is a CR that is used by software engineers to structure and monitor the progress of a software engineering process. Shared calendars, call tracking databases, and inter-office wikis are all examples of general CRs that can be used to share information and improve coordination within a multi-user environment.

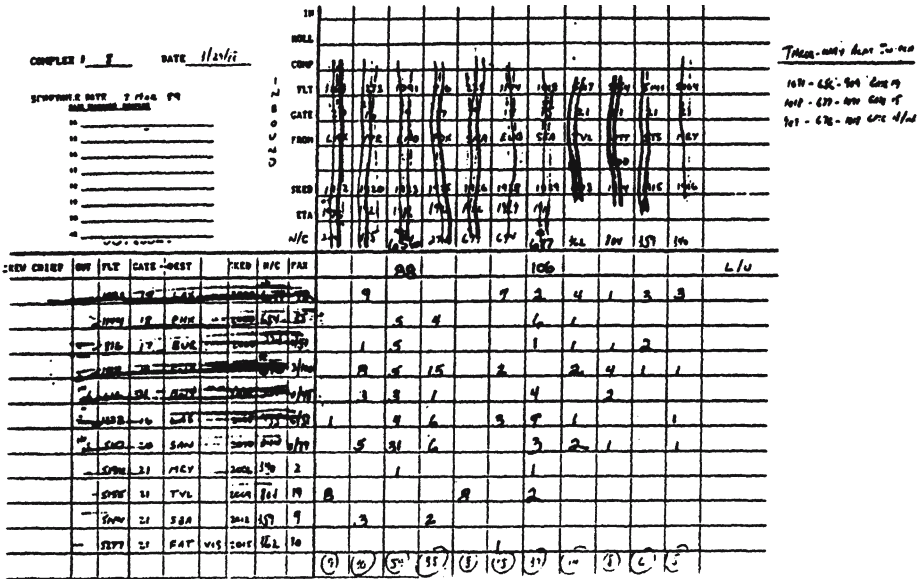

Fig. 1 The complex sheet (Suchman and Trigg 1991)

When CRs are introduced directly into a system's communication channels, we are able to enhance coordination both via the intrinsic usefulness of the CRs and also by gathering information that a groupware system can then use to provide intelligent support. This article discusses evidence of such expanded utility in one system and describes a methodology that can be applied to develop adaptive components in other groupware systems.

1.2 Organization of the argument

This work primarily concerned with how to acquire knowledge about users and their at runtime context to drive intelligent algorithms, without creating unnecessary or undesirable work for the users. The solution proposed is to modify the representational properties of the software system so as to both improve user coordination and gain access to the necessary runtime information. Hence, a single design solution is used to improve the system from the users' perspective, and from the perspective of the adaptive component designer.

To serve as a roadmap, an outline of the argument is provided here as a series of questions and answers.

1. **Question**: How can adaptive support be added to a groupware system?
   Answer: We infer users' intentions.
2. **Question**: Where can the run-time user information required for intent inference be obtained?
   Answer: This information is available in unstructured communication channels, but because it is unstructured it is difficult to access.
3. **Question**: How can this information be transformed so as to make it available to the system?

Answer: By introducing a structured communication channel in the form of a coordinating representation.

4. **Question**: If users are provided with this more highly structured communication channel, will they use it?
   Answer: Yes, because the structure that is provided by the coordinating representation makes it easier for users to exchange some kinds of information.
5. **Question**: Can a coordinating representation be developed that does not impair user performance?
   Answer: Yes; in fact, CRs improve domain performance.
6. **Question**: Is the information that becomes available via the use of the CR useful for intent inference?
   Answer: Yes, because it offers a window onto task and coordination-specific knowledge that is shared by users, and this knowledge is highly structured.

The results from case studies will be presented that confirm the answers to the above questions, demonstrating that:

1. Users choose to use the coordinating representation over chat to exchange some kinds of information (Sect. 4.1)
2. Using the coordinating representation helps users perform their task better. (Sect. 4.1)
3. The information provided by the coordinating representation enables intent inference with good accuracy. (Sect. 4.2)
4. An adaptive component that is driven by the information made available is heavily used, improves performance in the domain task, and reduces cognitive effort. (Sect. 5)

The article proceeds as follows. First, the domain, groupware platform, and associated intent inference technique are presented. Some of the representational deficiencies of the groupware system are then described; first from the perspective of the user, and then from the perspective of the developer of an adaptive component. It is then shown how a CR addresses these problems. Empirical data is presented that demonstrates how the CR benefits the users, and how information derived from its use improves intent inference. An adaptive component that uses information from the CR is described, and further empirical data confirming the utility of the adaptive component is presented. After a summary of the overall methodology, we provide a framework for comparing this methodology to other approaches to building adaptive systems. The article concludes with some final thoughts about the generality of the approach and future work.

2 VesselWorld

The experimental platform used in these studies is a groupware system called VesselWorld, in which three participants collaborate to remove toxic waste from a harbor. VesselWorld was demonstrated at CSCW 2000 (Landsman et al. 2001). The platform was designed with the goal of studying the coordination problems commonly faced by groupware users. The domain task requires varying degrees and types of coordination, collaborators have different roles and responsibilities, and coordinating information is exchanged via a chat window. These features are highly relevant to the study of groupware systems in general. VesselWorld proved to be very challenging for its users; in

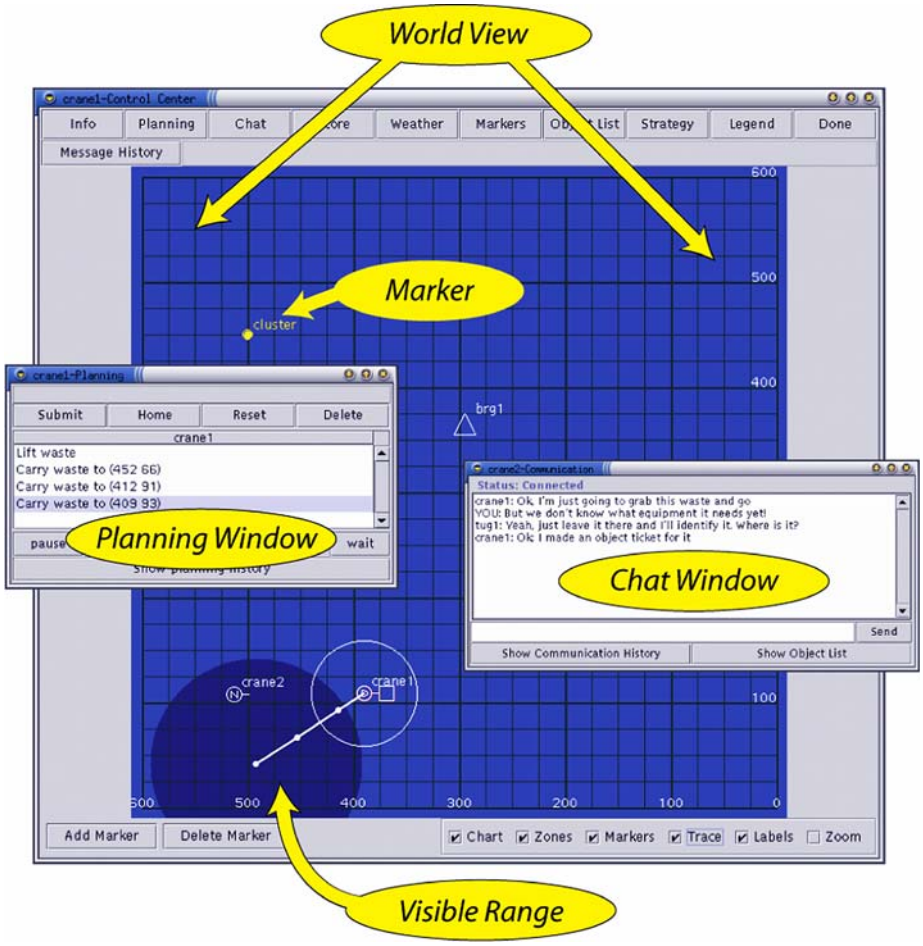

Fig. 2 The VesselWorld system

our case studies, we found that the performance of a given user group usually did not stabilize until after roughly seven hours of use (including a 2-hour training session).

VesselWorld presents a relaxed WYSIWIS environment, shown in Fig. 2, in which three participants play the role of ship's captains, and their joint goal is to remove toxic waste barrels from a harbor without spillage. The main interface ("World View" in the figure) is a shared map; the x (increasing to the west) and y-axes (increasing to the north) indicate latitude and longitude respectively. Users are provided a private "marker" facility so that they may annotate the world view with pertinent information (see the "Marker" in the figure). Each ship can only "see" a small radius around its current location (the darker circle marked "Visible Range" in the figure), so each user has different directly observable domain information; hence users must communicate to maintain shared information about wastes in the environment. Communication may occur at any point, but all communication occurs through a text-based chat window that is part of the system ("Chat Window" in the figure). VesselWorld logs

complete interaction data that can be used to replay user activity. This later feature is discussed further in Sect. 6.

The progression of a VesselWorld session is turn-based, such that every user must submit a single step to the server before it evaluates them and updates the world on each client screen. Users may plan any number of steps in advance, although each step can only involve objects that are currently visible. Plans can be managed (steps may be deleted or plans reset) via a separate planning window ("Planning Window" in the figure). Users' plans are not visible to each other, again requiring explicit communication to manage coordinated plans.

A VesselWorld session is complete when all toxic waste barrels have been moved to a large barge, which has a fixed position and unlimited capacity. Each ship has a different role in this process. Two of the ships have cranes that can be used to lift toxic waste barrels from the harbor and load them onto a barge. The third user is a tugboat that can be used to drag small barges (which have limited capacity) from one place to another. The crane operators can load multiple wastes onto the small barge, and at least one of them must also be present to unload the barrels and place them on the large barge. For notational convenience, we will occasionally refer to the crane operators as "cranes," the tugboat operator as the "tug," and toxic waste barrels as toxic wastes or simply wastes.

Wastes are of different types and require different coordination strategies to be removed from the harbor. A single crane may lift a small or medium waste, but two cranes must join together to lift and carry a large waste, and an extra large waste may be jointly lifted but can only be carried on a small barge by the tug. Wastes may require specialized equipment to be moved, and the cranes carry different types of equipment. The tug is the only actor who can determine the type of equipment a waste requires.

The users are scored by a function that takes into account the number of barrels cleared, the number of steps this took, the number of errors (dropped waste barrels) made, and the difficulty of the problem. In all user studies, the users were instructed to try to maximize their score.

2.1 Intent inference in VesselWorld

Planning in VesselWorld is a laborious and error prone operation (Alterman et al. 2001). User errors often occur because of forgotten plan steps or joint plans that have become unsynchronized. We sought to develop an automatic plan generation tool to address these problems. A hurdle in making such a tool useful is that there are an overwhelming number of potential goals for each user at any given time. Thus, an intent inference procedure was developed to reduce the number of possible goals to a manageable list from which users could then make a selection.

Bayesian Networks (BNs; Pearl 1988) were used to infer user intensions. BNs have been used in numerous systems and in a variety of ways to perform plan recognition (e.g., Charniak and Goldman 1993; Horvitz et al. 1998; Albrecht et al. 1998). The approach taken here is straightforward. The BNs used in VesselWorld are static and model the relationship between information about the state of the domain and users' likely intentions. At runtime, information about the domain is posted to these models for each agent-waste pair separately. The likelihood an agent has an intention with respect to a given waste is read off one of the nodes in the network, and the intention with the highest value for a given agent-waste pair is taken to be the most likely one

for that agent. New information is posted to the BN whenever a relevant change in the world is detected.

Two BNs were developed, one which infers crane intentions and one which infers tug intentions. Together, the two BNs can infer seven goal types; JOINT LIFT (⟨crane⟩⟨waste⟩), LIFT (⟨waste⟩), JOINT LOAD (⟨crane⟩⟨waste⟩), and LOAD (⟨waste⟩) for the cranes, and BRING (⟨small barge⟩⟨waste⟩), ID (⟨waste⟩), and TRANSFER (⟨small barge⟩⟨large barge⟩) for the tug. In this paper, we restrict our analysis to the portion of the crane network that predicts the Cranes' intentions to lift wastes. This BN is shown in Fig. 3; it models the likelihood that a crane operator has the intention to lift (or jointly lift with the other crane operator) a specific toxic waste.

The information used to determine if a crane has an intention to lift a waste is:

- The size of the waste (which determines whether a single crane can lift the waste, or the support of another crane is required). (*Node 1*)
- The type of equipment required by the waste and the type of equipment the crane has. (*Nodes 2 and 3*).
- Whether the cranes are close to or heading towards the waste. (*Nodes 4–7*).
- If the crane actor is currently holding a waste. (*Node 8*).

As portrayed in Fig. 3, the BN combines a procedural model of the task-domain with domain-specific heuristics that indicate which of those actions are most likely. For example, a procedural constraint is that an individual crane cannot lift a waste unless it either has the right type of equipment, or the waste does not require equipment; this information is captured in three nodes, "Need Equip," "Has Equip," and "Equip." These three nodes do not represent a simple rule, because the equipment requirements for a waste might not be known, and hence it is necessary to explicitly handle this kind of uncertainty.

Heuristic factors interact with the procedural model to differentially weight those wastes that are possible lift candidates. For the cranes, these factors are how close the ship is to a waste, how close the current heading of the ship is to a path that will intersect with a waste, and whether or not the operator is already holding an object.

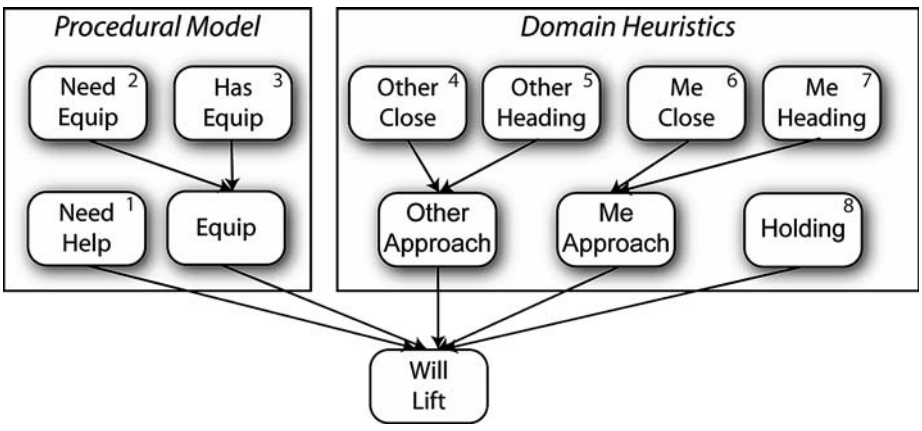

Fig. 3 Schematic of BN used to infer Crane Lift Intentions (information is posted to numbered nodes)

This heuristic information in part reflects the physical constraints of the domain; a ship must be close to a waste before it can be lifted.

As a whole, the BN models users that have a certain degree of proficiency with the domain. It is expected that such users understand the rules of the game, and will only try to lift wastes that they can. It is also expected that these users will proceed in a generally predictable fashion. That is, they will not zig-zag wildly upon an approach to a domain object, and they will not randomly pick up wastes and put them down again. It would be possible to incorporate additional models for less proficient users, or model users' domain concepts more precisely (e.g., Horvitz 1998). The focus of this work, however, is upon solving the difficulties in acquiring the user information necessary to drive the user model at runtime.

3 Representational problems in VesselWorld

Any piece of software may be considered to be part of a representational system. A representational system has three essential ingredients (Norman 1991):

1. The represented world (that which is to be represented);
2. The representing world (a set of symbols);
3. An interpreter (which includes procedures for operating upon the representation).

The representing world mediates the interaction between the interpreter and the represented world, and in so doing constrains the types of interaction that are possible between the two. As applied to software, the software system is a representational medium that constrains both how information in the represented world is viewed, and what procedures are available to the user for interacting with this world.

For adaptive systems, software can be seen to function as a mediating layer in two distinct but intersecting representational systems. First, and most clearly, it mediates the interaction between the user and domain (which may include other users), transforming the represented world so as to (hopefully) better support domain activity. The software system also mediates the interaction between the designer of the adaptive component and the user. In this later case, software very much constrains what information the designer can acquire about the user at runtime (St. Amant et al. 2003).

Here, representational deficiencies in VesselWorld are considered from both perspectives. The problem from the user's perspective, which is most traditionally thought of as an HCI problem, is considered first. The problem from the perspective of the adaptive component designer, which is a user knowledge acquisition problem, is then described. Both these kinds of problems are exactly those which Coordinating Representations can solve. Following the analysis of these problems, it is shown how the introduction of a CR in VesselWorld does just that.

3.1 Problems from the user's perspective

During a VesselWorld problem solving session, users must search for and discover information about the waste barrels that are floating in the harbor. Because each user has different locally available information, and recovering wastes requires the coordinated efforts of multiple users, it is necessary that participants communicate to establish mutual knowledge about the wastes. Managing and maintaining this shared

information comprises a significant portion of the work that users must do during a VesselWorld session.

VesselWorld initially provided two tools to support this work. The chat window allows users to communicate information about objects in the domain. It provides a scrollable history that records the entire dialogue during a given chat session, so that users can more easily recover information that may have been missed or is forgotten. The other tool is a "marker" facility, which allows individual users to annotate their own private maps (each user can only see their own markers). A marker consists of a highlighted point on the map, and a free-form text message for recording information (see Fig. 2).

These tools reflect the designers' initial vision of users' representational needs for managing domain information and establishing mutual knowledge about waste barrels. It was expected that users would publish local information via the chat window, and use the marker facility to record all waste information. During actual system use, it was found that these tools did not provide users with sufficient support. These representational deficiencies were most clearly manifest in specific features of the users' runtime dialogue.

One such feature was explicit talk about coordination, as shown in Fig. 4. In the example Crane2 suggests that the participants work out a shorthand for referring to waste information (line 3). Such explicit talk reveals a perceived need by the users for some form of coordinating structure.

Another feature of user chat was the emergence of explicit conversational patterns for managing domain information. For example, one group would perform a "marker check" when there were discrepancies among individual's local waste information, in which all of the known wastes would be validated. First, an actor would announce a marker check, and then proceed to transcribe all of her local waste information into the chat window. The other actors would confirm or contradict this information according to their own private views of the world. The transcription process that this group of actors undertook involved a significant amount of mundane work. However, users made the decision that this was less work than trying to recover from errors that were caused by inconsistent information about the world.

Recurring coordination problems were also indicative of a representational deficiency in the system. Frequently, one actor's reported perceptions raised questions from another actor, possibly about the size or location of a waste. This was usually caused by a lack of precision in communicating about locations, leading to difficulties in resolving references to objects. Sometimes, these errors would lead to one actor moving to the area of the waste in dispute to resolve the problem.

The coordination problems and dialogue features described above can be generally described as grounding problems (Clark 1996). The creation and maintenance of

Fig. 4 Explicit talk about coordination

```
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?
```

common ground forms a substantial portion of the work that goes into conversation (Fergusson et al. 1996; Traum 1994). Research has also demonstrated that technological media have impacts on grounding (Clark and Brennan 1990; Brennan 1998; see also Clark and Wilkes-Gibbs 1986).

To a degree, our analysis methodology (see Alterman et al. 2001; Feinman and Alterman 2003 for more detail) builds upon some of this earlier work on the grounding process. Our analysis elucidates those coordination problems that lead to the design of domain specific CRs, and grounding problems are one type of coordination problem that CRs can address. Critically, CRs solve two problems in the design of adaptive groupware. They address coordination problems for collaborating users, and they can also address a knowledge acquisition problem. In the following, the knowledge acquisition problems in VesselWorld are examined more carefully.

3.2 Problems from the adaptive component designer's perspective

The BN introduced in Sect. 2 is a model of the behavior the designer expects of users in different situations at runtime. For instance, the designer expects that if the two cranes are approaching a large waste and are not carrying anything, there is a good chance they will attempt to jointly lift the waste. In order for this model to function, information about the user and the user's runtime context is required. Some of this information can be derived from the plan steps users submit (where the users are, what direction they are heading in, what they are holding). However, a significant portion of the required information about wastes (where they are, what size they are, what equipment they require) is only available in the users' discussion about the domain. This information is very hard to extract automatically.

Figure 5 is a portion of the chat logs taken from a typical planning session in VesselWorld. Note that users sometimes refer to toxic wastes by coordinates on the shared map (e.g. "105, 420"). In the first line of the example, Crane2 announces a waste at (120, 420). In lines 2–4, Crane1 asks for clarification about the specifics of the waste. In lines 5–6, the tug replies (having apparently already investigated that toxic waste barrel) with corrected coordinates (105, 420) and specific information about the waste. In line 8, Crane2 thanks the Tug operator for the clarification, and the Tug closes the conversational turn in line 9.

This dialogue illustrates some of the problems in automatically extracting the domain information required as input to our intent inference procedure. In order to associate information appearing in separate utterances with a single concrete waste,

Fig. 5 Excerpt from chat during VesselWorld session

```
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
```

it is necessary to correctly resolve references. However, because the dialogue occurs between three active participants, the conversational turns that might be used to narrow the reference resolution scope are hard to identify. Furthermore, referring expressions can change from utterance to utterance even within the same conversational turn. For example, line 1 refers to the waste as "120 420" and line 5 refers to the same waste as "105 420." People can sometimes handle such ambiguities, but this is problematic for automatic reference resolution algorithms.

Within user groups, some structural conventions for referring to domain objects do emerge, but the type and use of this structure varies widely between and within groups. Some groups are explicit in defining a shorthand; other groups converge to common conventions over time. Within groups, referential styles vary between participants and over time. Oftentimes, multiple conventions will co-exist. In Fig. 6, a section of a transcript is shown for a group that exhibits this behavior. 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 and the location portions 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 the tug places a type description after the location. The participants never converge to a common convention.

For purposes of knowledge acquisition, 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. Rather than developing specialized algorithms to deal with the nuances of three-way, live chat in the VesselWorld domain, it would vastly simplify our task if users were to enter all the information the system needs in a common, structured form. Although this might seem like it would unnecessarily burden the user, below we will see that this is not the case.

4 Using CRs to fix representational problems

VesselWorld has representational deficiencies from the both the user's and the adaptive component designer's perspectives. In mediating the interaction between the users and domain, it does not offer the right kind of support for managing information about wastes. In mediating the interaction between the designer and the users, it does not provide structured access to the runtime information required for successful intent inference. The analyses performed above led to the design of a Coordinating Representation, which is a solution to both of these problems.

There are other approaches to designing of collaborative artifacts that address various issues in collaboration. Cognitive work analysis (Vincente 1999) advocates a host

Fig. 6 Different ways of referring to wastes, within one group

```
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
```

of ethnographic methods for developing interfaces that address collaborating users' needs. Activity theorists focus on the need for mediating artifacts that address tensions in the larger activity system (e.g. Engeström 2000). Suthers (2003) approaches the design problem from the perspective of the teacher, as a way to improve the process of collaborative inquiry. However, no existing framework draws together design problems from the perspectives of both the user and the adaptive component designer.

Several CRs were developed for VesselWorld (Alterman et al. 2001). The CR that addresses the representational problems described above is called the Object List (Fig. 7). The Object List is a WYSIWIS (What You See Is What I See) component that helps users to manage domain information and coordinate references. 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 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 first on the field and then on the object on the map. 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 World View (Fig. 2) as icons that are annotated with the name that is in the "Name" field.

There is nothing exotic in the design of the Object List. It is remarkable as the product of a repeatable design methodology that is explicit guidance for the creation of shared artifacts which simplify coordination for users and make intent inference easier. In the following, empirical data is provided that demonstrates the Object List indeed achieves both of these goals.

4.1 Improvements from the user's perspective

In all user experiments, the Object List was heavily used, and the coordination problems described above were no longer observed. The Object List had significant qualitative impacts upon the way users coordinated information about toxic waste barrels. Figure 8 compares sample dialogues from users who did not have access to the Object List (the left column) to those that did (the right column). It is immediately apparent that the group that had the Object List spent far less time talking about the details of

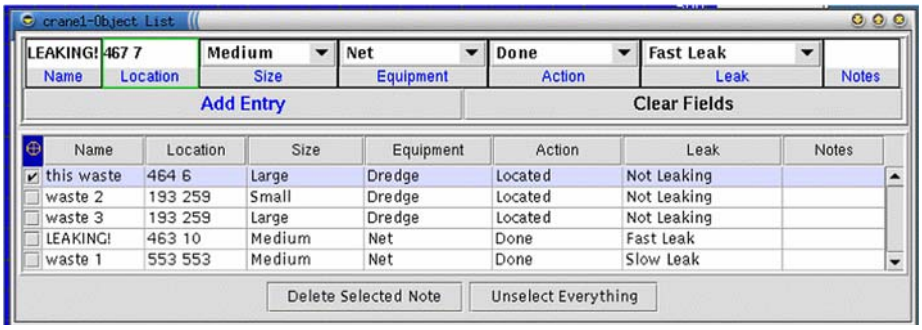

Fig. 7 The object list

Without the Object List	*With the Object List*
1. Crane2: what was small at 275, 400?	1. Crane1: I got an XL!
2. Tug1: sX	2. Tug1: I got nothing, you luck basrstartd.
3. Crane2:ahve sX at 450 above that	3. Crane2: I got an Xl and an L, mommy! ;)
4. Tug1: mD at 400 350	4. Tug1: Merry christmas kids….
5. Tug1: yes, there is an sX at 275 250 as well	5. Crane1: I'll map North third?
6. Crane2:I have two amall at 275, 400 and 275, 450 are these the same?	6. Tug1: I'll take middle 3rd.
7. Tug1: no, there are two sX there	7. Crane2: I'm at south-central. Tug, where are you?
8. Tug1: well, there are actually three in almost a line from n-s	8. Tug1: I'm jus nw of the barge, let me put that on the map…
9. Crane2:at 400 and 450? what about the 275, 250?	9. Tug1: actually more w than n.
10.Crane2:ok, so the southern exists?	10.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
11.Crane2:I'm almost there, nothing at 275, 250	11.Crane1: not a bad plan…
12.Tug1: 300 350, 250 400, and 275 250 are my markers	12.Tug1: Ok, I'll make a bit of a sweep around here while CRANE1: looks around.
13.Tug1: argh	13.Crane1: Tug, can you pick up the SB at your earlier opp?
14.Tug1: I mean 275 450	14.Tug1: CRANE2: can map up on the way?
15.Crane2:ok, those sound good	
16.Tug1: i don't know why I kee doing that.	

Fig. 8 Dialogue from users before an after the introduction of the object list

⊕	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	

Fig. 9 Object List created during dialogue in Fig. 8

each waste. None of their discussion appears to be designed to exchange information about the wastes per se, but rather is focused on strategizing.

Figure 9 depicts the information users entered into the Object List during the dialog in the right hand column of Fig. 8. This demonstrates that the information previously exchanged via the chat tool has now been offloaded into the Object List.

To assess the impact of coordinating representations quantitatively, a formal study was performed comparing the performance of teams with and without CRs (Alterman et al. 2001). Each test group consisted of three teams of three subjects. Subjects were a mix of area professionals, mostly in computer-related industries, and undergraduate students; all were paid a flat fee for the experiment.

The results of this study, shown in Table 1, compare the final 5 h of play for each group (by which time performance had stabilized), and these results were normalized over a general measure of complexity for each of the problems solved. The perfor-

Table 1 Improvement of CR groups over non-CR groups; final 5 h of play

Indicator	Improvement (reduction)
Communication	57% ($p < 0.01$)
Domain errors	61% ($p < 0.2$)
System events	38% ($p < 0.06$)
Clock time	49% ($p < 0.01$)

mance of the test group that had the CRs was better across several measures: amount of chat, number of errors committed, number of system events generated (an indicator of interface work), and clock time.

The most significant effect is the 57% reduction in communication generated. This confirms the observation 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%. Additionally, 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 overall low frequency of this kind of error; this reduced its confidence below statistical significance ($p < 0.2$).

The above results demonstrate that the introduction of the Object List modifies the representational system from the user's perspective in ways that improve the ability of the collaborators to coordinate information. It is shown in the following how the Object List also addresses the representational needs of the designer.

4.2 Improvements from the adaptive component designer's perspective

The Object List captures information about the world that would otherwise be exchanged in chat, and transforms it into a form that can be easily used in the intent inference process described above. In addition to information about where the wastes are, what equipment they need, and what size they are, the Object List also provides the system with a set of user-assigned labels. These labels can be used to mine chat for additional information about users' intentions. Detailed information about how chat information was used can be found in Appendix A.

The relative utility of information from the Object List for performing intent inference is evaluated in the following. This is done by comparing the ability of the BN to predict user lift actions in historical log files across several information conditions. Before presenting results, the evaluation methodology is described in detail.

4.2.1 Evaluation methodology

To evaluate the utility of information from the Object List, the BN was trained and tested under four information conditions using a single dataset (described in Table 2), and its performance was compared across these conditions. The dataset consists of time-stamped information derived from the log files collected from each use session in the dataset. It contains the state information required by the BN. State information includes information about each waste (position, size, and equipment needs), the users' locations and heading, and wastes as they are lifted. The dataset also contains information about references made by users during chat to objects that are listed in the Object List.

Table 2 Dataset for the evaluations

Team	Sessions	Avg. # of wastes per problem	Total Hours
Team 1	10	11.7	9.9
Team 2	6	11	8.4
Team 3	9	14.3	9.1
Team 4	16	14.5	8.7
All	41	13.5	34.3

Each condition changes the amount and quality of information about wastes in the dataset. The four information conditions, and the information available under each condition, were:

1. *Perfect Information* – Under this condition, the dataset contains complete and accurate information about the position, size, and equipment needs for every waste at each point in time. Perfect information is not typically available at runtime, but it can be derived from the problem files which define the initial problem configuration for a usage session. This is an ideal case, and provides a baseline against which to compare the other information conditions.
2. *Object List* – Under this condition, the dataset contains information about the position, size, and equipment needs for wastes that are entered in the Object List, as it becomes available. This information is subject to errors, duplication, and omissions. Compared to the Perfect Information condition, performance of the BN under this condition is indicative of the quality of the information users provide via the Object List.
3. *Perfect info + Chat* – All of the information in the Perfect Information condition, plus the occurrence of names of objects (as entered by users into the Object List) in chat. More precisely, there is an entry in the dataset for each time a word appears in chat matches one of the labels for a waste in the Object List (see Appendix A). This condition is used to investigate how much predictive information references to wastes in chat add.
4. *Object List + Chat* – Information from the Object List condition, plus the occurrence of names for objects in chat. This condition represents the best possible runtime performance of our inference procedure without information from problem files.

Because the dataset contains different information under each condition, the probability distribution stored in the BN may be optimal for one information condition but not another. If the evaluation were to be performed using a single probability distribution for all data conditions, results might reflect the "preference" of that particular probability distribution, rather than the utility of information contained in the dataset.

To overcome this problem, the probability distribution of the BN was fit to the dataset in each information condition before testing. To train the network, each dataset is compiled into a series of cases. A case consists of values for each non-hidden node in the BN (see Sect. 2.1), and each case represents a change in the relevant information about the world. During the training phase, values for the "Will Lift" node were extended back in the dataset from each actual lift for a 10 min time window. The size of the time window was based on initial experimentation, and chosen to maximize the performance of the BN.

The $EM(\eta)$ algorithm (Bauer et al. 1997) was used to train the network, and the same starting parameters for the BN were used in each case. The $EM(\eta)$ algorithm is a generalized version of the standard EM algorithm with a learning parameter. When $1.0 < \eta < 2.0$, the $EM(\eta)$ algorithm is significantly faster than the standard EM algorithm, but still has good convergence properties. For all of our training sessions, $\eta = 1.8$. Algorithm performance was further optimized by posting only unique cases to the network, and weighting each unique case in the training set according to the number of times it occurred.

For the evaluation, two performance metrics were calculated: the correct goal rate (CGR), which is the proportion of correctly guessed goals; and the false positive rate (FPR), which is the proportion of guesses that were false. A guess is made whenever a relevant state variable changes. Any uninterrupted sequence of correct guesses leading up to the step immediately preceding the execution of the predicted goal is counted as a single correct goal. The total number of goals is the number of wastes lifted. Thus,

$$\text{CGR} = \text{correct goals/total goals}$$
$$\text{FPR} = \text{incorrect guesses/total guesses}$$

4.2.2 Evaluation

The evaluation was performed solely to compare the utility of different information sources for intent inference, and in particular, the utility of the information derived from the object list. These performance metrics are not indicative of how well the intent inference procedure performs within the context of a specific adaptation. Studies of our adaptive component are documented in the next section.

The results of the evaluation are shown in Table 3, and reflect average performance across all teams for each of the four data conditions. A two-factor ANOVA was used to evaluate the effects of the two variables (Perfect Info vs. Object List, and with or without Chat information) for both CGR and FPR. The analysis revealed no significant interactions between the variables. For the CGR, the effects of both variables were significant, but for the FPR, only performance differences between the Perfect Info and Object List conditions were significant. These results were corroborated via further analysis using paired t-tests. In general, this analysis indicates that the algorithm performs better when perfect information is available, and that chat information reliably increases the number of lift actions correctly inferred. Chat information does not, however, reliably reduce false positives given this training and testing methodology.

Table 3 Intent inference results for different information sources

Correct goal rate	With chat	Without chat	*p-value*
Perfect Info	0.87	0.83	$p < 0.03$
Object List	0.77	0.70	
p-value	$p < 0.001$		
False positive rate			
Perfect Info	0.51	0.53	$p = 0.4$
Object List	0.58	0.60	
p-value	$p < 0.001$		

The "Perfect Info" case, in the top row of the table, provides a baseline against which results for the other conditions may be compared. It is a rough indicator of the best the intent inference procedure can do, given only complete information about the state of the world. Across the four teams in the dataset, the Correct Goal Rate for the "Perfect Info" case ranged from 0.77 to 0.91 for this condition. There was a weak correlation between problem size (number of toxic wastes) and individual team performance ($r = 0.23$), reflecting the fact that it is more difficult to make good guesses when there are more options to choose from. In general, these metrics indicate that the inference procedure is effective.

As expected, the intent inference procedure does not perform as well with information from the Object List alone. However, results from the "Object List" condition were still good, and demonstrate that use of the Object List was reliable enough to be useful for intent inference.

The "Perfect Info + Chat" condition demonstrates that references add significant information that cannot be derived from knowledge about the state of the domain. Thus, regardless of access to state information (for instance, if there were intelligent sensors placed in the world) the Object List adds information that still improves intent inference.

The combination of reference information from chat and domain information from the Object List (the "Object List + Chat" condition) improves the performance of the procedure to a point where it is nearly as good as the "Perfect Info" condition. These results confirm that, by modifying the representational properties of the media between the user and adaptive component designer, high quality runtime information can be made available for intent inference.

To demonstrate that this information is sufficient to drive a useful adaptive support mechanism at runtime, an adaptive component was implemented and its use evaluated. These results are described in the following section.

5 An adaptive component

As described in Sect. 2.1, planning in VesselWorld is a laborious and error prone operation, and user errors are frequently the result of forgotten plan steps or joint plans that have become unsynchronized. The adaptive component was thus designed to help users formulate basic individual and joint plans. Using the intent inference procedure, the system can provide a small set of likely plans from the hundred or so that are possible at any point in time.

The interface to the component is shown in Fig. 10. It displays inferred plausible domain goals for each of the users. Like the Object List, it is a WYSIWIS component, so each user can see the other users' goals as well. Users can select plausible goals from their own column to have the system automatically generate a plan. The component can generate individual as well as shared plans. If the information in the Object List is correct, the generated plans are guaranteed to be correct.

The detailed function of the adaptive component is as follows:

1. Each time state information is updated (e.g., when plans are executed, information is added to the Object List, an object reference appears in chat, etc.) the system offers each user up to five plausible goals, displayed in order of priority and color-coded according to the system's "confidence" in that prediction. Each user can only select goals from their column.

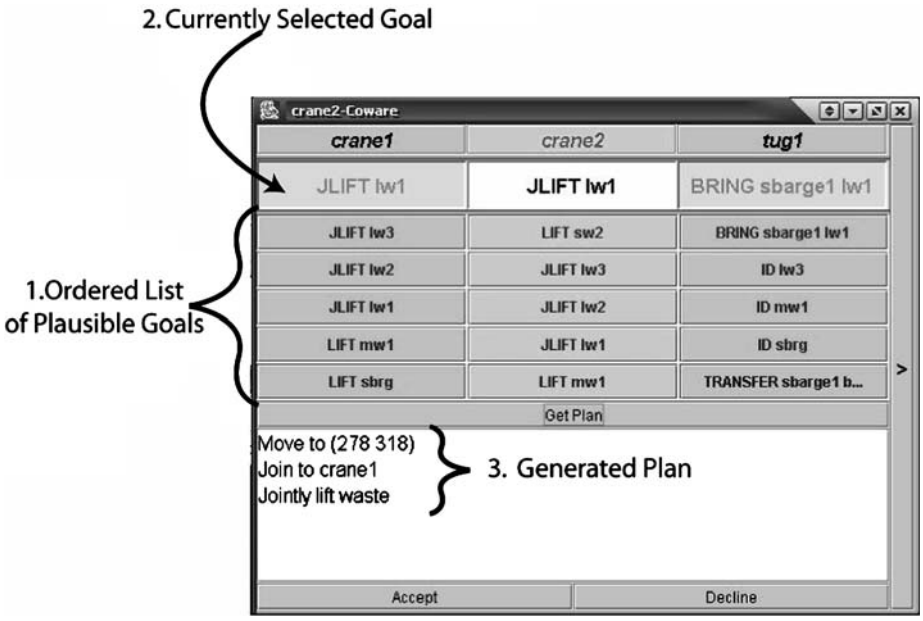

Fig. 10 The adaptive component

2. When the user selects a goal, it is copied into the top row, which displays each user's currently confirmed goal. The user then has the option to request an automatically generated plan that will accomplish the selected goal (the "Get Plan" button shown in Fig. 10).
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 accept the invitation, a plan is generated; if invited actors do not accept the invitation, the requesting user is so informed.
4. If the user accepts the plan (by clicking the "Accept" button in Fig. 10), it is automatically copied into the user's planning window for execution.

If the plan is generated from correct state information, no user modifies the state in such a way that conflicts with the generated plan, and the plan is executed to completion, it will succeed in achieving the desired goal.

5.1 User studies

We performed a formal study with four teams to evaluate the use of the adaptive component and its effects on performance. Two of the teams were given access to the adaptive component, and the others were not. The participants were a mix of students and local-area professionals, with varying degrees of computer proficiency. Each team was trained for 2 h in use of the system, and then solved randomly chosen VesselWorld problems for approximately 10 h. To alleviate fatigue concerns, the experiment was split into four 3-h sessions.

For the teams with the component, the inference procedure used information from the Object List and chat (the Object List + Chat condition) to infer user goals and

construct plans. The following results report on the last 5 h of play time for each group, by which time performance of the users had stabilized.

5.1.1 The component was heavily used

All groups used the component to generate plans within the system. On average, users confirmed a goal every 1.5 min (SD = 46 s), requested a plan for each confirmed goal, accepted 71% of plans requested (SD = 19%), and completed the execution of 83% (SD = 6.75%) of these plans. Overall, this indicates that roughly 59% of confirmed goals resulted in a plan that was executed to completion. For each problem solving session, one quarter of all plan steps submitted to the server were generated by the component (SD = 8%).

The component generated plans for 43% (SD = 15%) of the domain goals it could have predicted for the Cranes (some goals, like "search the harbor" were not inferred). It was not possible to obtain a similar statistic for the Tug operator because it is difficult to recognize the tug's goals in the collected log files (goals for the tug are not bracketed by easy to detect plan steps like "LIFT" and "LOAD").

5.1.2 The component improved users' performance in the domain task

The groups that had the component had 45% ($p < 0.10$) fewer joint errors (failures during joint actions) per minute than the groups that did not. This difference is not significant at the .05 level because of the small sample size and overall low proportion of joint errors. A reduction in joint errors corroborates prior analysis of use of the VesselWorld system, which indicated that joint errors were usually the result of plan submissions becoming unsynchronized (Alterman et al. 2001). Because the component generates coordinated plans in advance, users could simply submit each step and be assured that actions would be coordinated.

5.1.3 The component reduced cognitive effort

To measure the change in cognitive effort between the two populations, the amount of time it took users to execute plans and the amount of interface work were evaluated. It was found that the amount of clock time taken by users between submitting steps of automatically generated plans was 57% less ($p < 0.01$) than in groups without the adaptive component, but that there were no significant differences in the number of mouse clicks per waste. Because the reduction in clock time for groups with the component cannot be explained by a reduction in the amount of interface work, we conclude that the component reduced the cognitive effort of the collaborators.

As stated in Sect. 1.2, these studies show that the adaptive component was heavily used, improved user performance, and reduced cognitive effort during plan execution. This is verification that collaborating users can generate enough structured information when using a coordinating representation to drive useful intelligent support. We conclude that the approach to adding intelligent support demonstrated here was successful for this domain.

6 Summary of the methodology

This work has demonstrated how the design of a groupware system can be modified to address the needs of two intersecting representational systems. This is done by adding structure to the system in the form of a Coordinating Representation. The CR helps users coordinate and improves performance in the domain task. It also renders information previously exchanged via an unstructured communication channel accessible for use in an adaptive support system.

Clearly, much in this approach rests on the design of the CR. To facilitate the application of the approach to other domains, a repeatable design methodology for CRs with the necessary properties has been developed (Alterman et al. 2001, Landsman and Alterman 2005; Feinman and Alterman 2003). This methodology can be summarized in four steps:

1. Transcripts are collected of runtime user behavior.
2. These transcripts are analyzed in order to identify weak spots in the user's representational system.
3. This analysis is used to develop coordinating representations that people will use, and that improve coordination.
4. Information collected by the CRs is leveraged to drive adaptive components.

Transcript collection has roots in ethnography and interaction analysis. Traditionally, ethnography has relied upon *in situ* observation. Suchman and Trigg (1991) and Goodwin and Goodwin (1996) explored the use of video recordings in capturing and analyzing naturally occurring workplace activity. While video is a very powerful technique for studying a work practice, it is a resource intensive process. For the ethnographer, groupware makes life much easier, because a large portion of the relevant activity is mediated by the system itself, and can thus be passively recorded and analyzed offline.

To perform the kind of ethnographic analysis required to identify representational deficiencies in groupware, it is helpful to be able to replay transcripts, rewind when necessary, easily locate high-level task events (e.g., the submission of plans), and add annotations. To provide this functionality, a component-based software framework called THYME was developed (Landsman and Alterman 2005). Groupware systems built using THYME automatically log all usage data, and replay tools can be rapidly generated. The replay tools provide a console similar to that on a VCR, but with all of the above features.

Two techniques have been developed to analyze transcripts. One of them, which was described briefly in Sect. 3.1, examines users' recurrent activities to identify weak spots in an existing representational system (Alterman et al. 2001). The other technique examines co-reference chains (Feinman and Alterman 2003). To perform this later analysis, the analyst identifies and tags all references to objects, plans, and other entities of interest in a logged dialogue. Metrics are then calculated for the distribution and lifespan of each type of reference and these metrics are used to identify the need for supporting structure. Empirical evidence has been collected demonstrating that these analysis techniques can be used to make predictions about the type of structure that will be useful (Feinman 2006).

These analyses guide the development of CRs that address representational needs of both the user and the designer. Because CRs fix problems with an existing representational system for the users, they will improve the users' ability to stay coordinated,

and users will use them. This article has demonstrated how the structure introduced by CRs can than be leveraged to introduce adaptive support.

7 Balancing user effort and adaptive component designer effort

As discussed above, adaptive software can be viewed as the mediating layer in two intersecting representational systems. This situation is depicted in Fig. 11. In the diagram, the software system is presented as a composition of three components. This is intended as a descriptive conceptual breakdown only, and is based loosely upon early adaptive software models that grew out of user-interface management systems (e.g., Hefley and Murray 1993).

The three components of this model are:

The interface—Contains the signals and controls that the user interacts with directly. It is analogous to Norman's (1991) surface representation.
The domain model—The system's internal representation of the task-domain. It defines the functionality of the system. The domain model is very often implicit in the code that makes up a system, rather than represented concretely.
The adaptive component—May contain a user model, domain knowledge, an inference engine, and other technology related to run-time adaptation. It serves as a proxy that supports the time delayed interaction between the designer and user. It encapsulates the designer's assumptions about the user's potential needs at runtime, what information is needed to recognize those needs, and how they should be met.

There are two representational systems in the diagram. In one, software mediates the interaction between the user and the domain; this is the user's perspective. In the other, the software mediates the interaction between the user and designer of the adaptive component; this is the designer's perspective. This conceptualization highlights what are often considered competing design requirements for adaptive systems (Schneiderman and Maes 1997). The choices that are made with respect to the balance between these requirements are reflected in the design of the interface. These

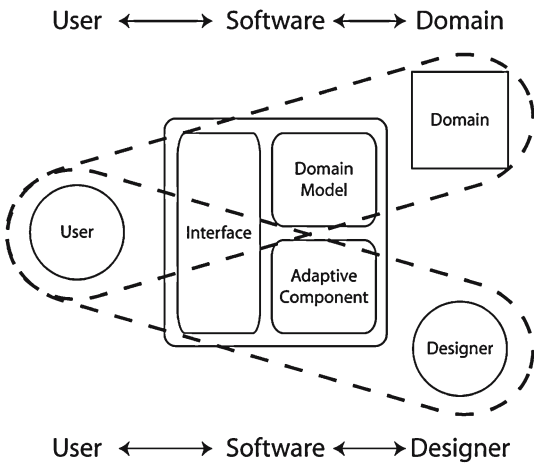

Fig. 11 Two intersecting representational systems

choices have consequences for the amount of work the user must do, and the resources expended in developing the system.

Several other systems are examined here with respect to the balance struck between these competing design requirements. This balance is characterized by the degree to which specific interface features are incorporated to meet the design requirements from either perspective. The set of features meeting design criteria from the user's perspective are referred to as a *task-language*. A task-language is devoted to supporting user's manipulation and interpretation of the domain-task. In VesselWorld, a user plans a step to move his ship closer to a waste using the task language.

The set of features meeting the designer's criteria (from the perspective of the intent inference engine) are referred to as a *meta-language*. A meta-language is not the task-language itself, but is rather a set of interface features that allow the user to tell the system *about* the task being performed. VesselWorld does not have a meta-language (c.f. Alterman 2000).

7.1 Task languages

Task-languages are designed to mediate and support user action in a domain. Examples of task-languages include direct manipulation and command-line interfaces. In the case of groupware, a chat window is part of the task-language, because it mediates the user's coordination with other users (part of the domain task). Some user-adaptive systems infer user's needs by monitoring their task interaction alone. These systems contain few or no meta-language operators (Fig. 12).

In such systems, the feasibility of and effort involved in developing the adaptive component depend upon how easily interaction data can be interpreted against the backdrop of available task and domain knowledge. For example, the SmartEDIT system (Lau et al. 2000) is a programming by demonstration system that can infer user plans for making syntactical edits to a document. The system does employ a minimal meta-language that requires the user to segment the execution trace into training examples, but its inference mechanism interprets task-level information directly.

SmartEDIT's inference mechanism (a version space algorithm) is successful for two reasons. All relevant user information is readily available by monitoring changes to the text buffer, the cursor location, and the clipboard contents. Furthermore, a set of domain operators that cover a large portion of the syntax editing task-domain are

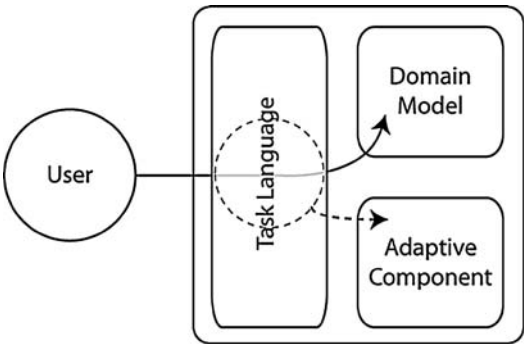

Fig. 12 Adaptive system with task-language

readily specified. For such well-defined domains, the costs associated with knowledge engineering can be relatively small.

Recommender systems are a different class of user-adaptive software that require minimal or no meta-languages. Among the most widely used of these (e.g. Amazon.com; Google), knowledge acquisition depends only upon a user's task level behavior—browsing, purchasing, or searching. The two most commonly used inference techniques in widely deployed recommender systems are collaborative filtering (Goldberg et al. 1992) and text-based information retrieval. For collaborative filtering, domain knowledge is based upon statistical patterns of many users performing similar (browsing, purchasing, or searching) behaviors. Text-based information retrieval depends upon statistical patterns of words in a large corpus of documents that others have created. In both cases, the bulk of the knowledge engineering effort is offloaded to external sources.

Generally then, for adaptive systems that employ a task-language:

User effort to engage the adaptive component is low. The user is willing to use the interface; this is the sole interface through which the user interacts with the domain. This does not imply that the system is easy to use; it only means that knowledge acquisition does not introduce much extra work for the user.

Adaptive component design effort is domain dependent. Often, importing a sufficient amount of domain knowledge into the system requires a significant investment of engineering effort, though sometimes this domain knowledge is available from external sources.

7.2 Meta languages

When the system cannot extract enough information from the user's interaction with a task-language to support adaptation, a meta-language can be included. Examples of meta-languages include preference dialogs used to configure a system's behavior, or a natural language interface to an agent. The defining feature of a meta-language is that its primary role is to address the knowledge acquisition needs of the designer (Fig. 13).

Meta-languages cover the spectrum from formal to natural, and may support a more or less objectified interaction with the adaptive component. The Collagen system (Rich and Sidner 1998; Lesh et al. 1999) is an example of a system that used a formal meta-language to support an objectified interaction with an agent. Collagen is a middleware framework for the development of collaborative agents that can

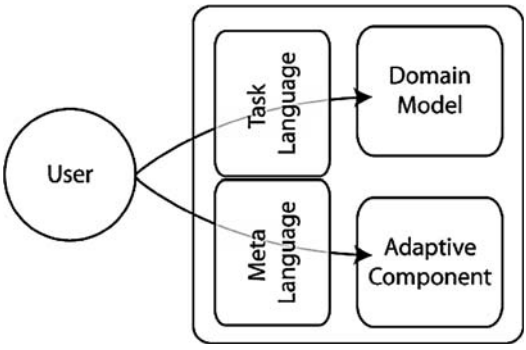

Fig. 13 Adaptive system with meta-language

provide guidance and offload some user work in the domain task. In its initial formulation, Collagen included a formal meta-language called ADL (Artificial Discourse Language) (Sidner 1994). The user provided the system with information about her goals via a menu-driven interface to ADL (which provided a list of valid dialogue moves), and the intelligent component then used this information to provide support in the form of suggestions about next steps and contextual reminders.

In Collagen, ADL reduces development costs in two ways. It provides the system with structured information about the user's intentions. It also enforces a particular interaction between the user and adaptive component, so that domain knowledge and the algorithms associated with its use can be represented in a portable and domain independent way. In this manner, Collagen forms an adaptive systems framework that can greatly reduce development costs in the long-term.

A drawback with the formal/objectified approach is that the user is required to manage two tasks instead of one. One is her desired domain task, and the other is explaining her actions to the system. This second task is made even more cumbersome when it requires the use of a formal language. As was noted by Collagen's developers, "it is often more efficient and natural to convey intentions by performing actions" (Lesh et al. 1999; p. 23).

One way to avoid requiring that the user manage two tasks is to combine the meta-language with the task-language, so that the interaction no longer objectifies the adaptive component and the user is free to focus on the domain task. For example, the Epsilon collaborative learning environment requires interlocutors to use sentence openers that represent speech acts during chat, and this structure is in turn used to identify problems with the collaborative process (Soller 2004). The task and meta-language are coincident in Epsilon, but still distinct, as the structure that is added to the task-language is done so solely to address designer's knowledge acquisition needs. This kind of approach can lead to interference with the user's domain-task.

Natural meta-languages (which rely upon natural language understanding) can be employed to avoid problems with formal languages; they provide a rich source of information about the user, and potentially require little or no training to use. Interaction via a natural meta-language usually requires an objectified adaptive component (a conversational "agent"), which potentially raises the same "two-task" problem as above. However, sufficiently advanced natural language would allow the user to choose the best modality for the task at hand without ever having to focus on the meta-task of making herself understood.

The major drawback with natural language understanding is that technology is not yet advanced enough to scale beyond special purpose applications (Zukerman and Litman 2001). This results in additional work from both the user and designer's perspective. To the user, the interface may appear to support a wider range of conversational moves and vocabulary than it actually does, and the user is left to navigate a dialog through trial and error (Schneiderman 1998). For the designer, a good deal of effort must go into building the language recognition facility, and input to the intelligent component cannot be easily constrained, and so may be more difficult to process.

Numerous approaches have been developed to manage the tradeoffs when employing meta-languages. For example:

- Mixed initiative systems, such as Lookout (Horvitz 1999) try to calculate the tradeoff between the benefit provided by the adaptive system and the additional work

the meta-language requires from the user. An assumption in this approach is that at certain points, the user may *want* to engage an intelligent agent, and that this can be recognized with a sophisticated enough domain task model and processing of the task-language.

- Some systems adapt the meta-language itself over time by allowing the user to introduce or redefine terms. For instance Pan et al. (2005) describe a system in which the system and the user adapt to one another to achieve a common vocabulary. This form of two-way adaptation is an interesting take on end-user-programming (Nardi 1993), and may lower the bar for casual users that would not otherwise be interested in customizing the system's behavior.
- Multi-modal systems incorporate other input modalities along with limited natural language processing to improve system comprehension. The CASIS system (Leong et al. 2005) uses contextual cues like room temperature, ambient noise, and room activity to reduce error rate in speech recognition. Such systems are still in early phases of research, and require specialized hardware in controlled settings, but offer significant performance gains over mono-modal meta-languages.

Generally then, for adaptive systems that employ a meta-language:

User effort may be too high. Whether or not people are willing to use a meta-language depends on the potential benefit and the design of the language. Using a meta-language can disrupt the domain task, either by requiring the user to manage an additional task, or by interfering with the task-language. Meta-languages can trade reduced design costs for additional user effort.

Adaptive component design effort can be reduced by implementing a meta-language, although this depends on the type of language used. Often, a designer's efforts to make a meta-language more appealing to the users, as with the case of natural languages, ultimately results in more overall work (for both the designer and the user) than the meta-language saves in terms of making inference easier.

7.3 Coordinating representations: combining task and meta languages

The approach described in this article employs a coordinating representation to serve the needs of both the designer and the user. The task and meta-languages are not merely coincident (as with the Epsilon system above). Instead, the CR combines the task-language and meta-language into a single interface mechanism that serves both roles (Fig. 14).

The general idea of combining user needs and designer needs into a common interface tool is not restricted to collaborative systems (see St. Amant et al. 2003, for a discussion). However, as we have shown, groupware is particularly amenable to the approach because of the difficulty that users have in coordinating their activities. For VesselWorld, it was precisely the difficulty in maintaining common ground that makes the structure of the Object List so useful for users. This structure is equally useful from the designer's perspective, because it renders the users' task-relevant common ground accessible to runtime inference mechanisms.

There are other systems that use coordinating representations in a similar manner, although their role is rarely given such a central focus. Jameson et al. (2003) describes a collaborative multi-agent system for scheduling a business trip, in which agents serve as proxies for users in a scheduling scenario. The role of the agent, and one of the main foci of that work, is to provide a social face to help users stay aware of others' preferences and motivations. However, most of the information that is derived from

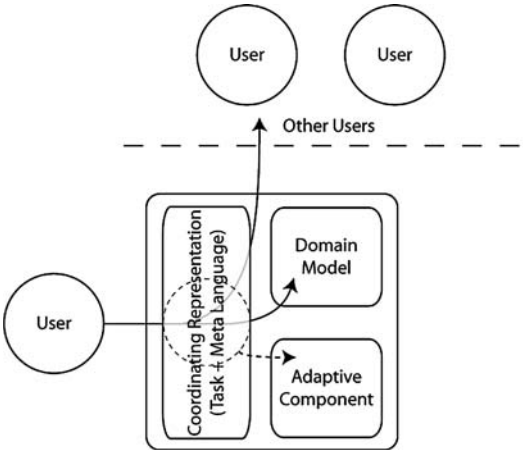

Fig. 14 Adding structure to a task language to enable intelligent support in groupware

the users comes from a coordinating representation that allows them to express their preferences and explore possible solutions to the constraint problem set up by the collaborative task.

Concept maps are a class of CRs that have proved useful in adaptive systems. Concept maps are visual representations of interlinked concepts that can be useful for people tying to communicate about large conceptual structures or coordinate knowledge engineering efforts (Novak and Gowin 1984). Cañas and Carvalho (2004) describes several adaptive systems that use concept maps in collaborative settings. One system uses the structure in a concept map, along with WordNet (Miller 1990) to disambiguate word senses. Another system draws conclusions from the concept maps shared by students in a collaborative learning environment. These conclusions can help the students to identify weaknesses or inconsistencies in their own conceptual structures.

Generally then, for adaptive systems that employ coordinating representations:

User effort is reduced. Communicating coordinating information, such as required in maintaining common ground, is part of the domain task in a groupware system. As set forth in Sect. 4.1, our empirical data proves that CRs can make this easier, and because CRs actually reduce user workload, users are willing to use them.

Adaptive component design effort is low. Information can be easily leveraged for adaptation because it is highly structured, and germane to the users' joint task. Our approach in VesselWorld used widely available AI techniques, and the data discussed in Sect. 5.1 validates the effectiveness of the resultant adaptive system.

8 Generality and future work

Coordinating representations offer a means for merging two traditionally competing design requirements for user-adapted systems into a single representational medium. On the one hand, CRs can reduce user workload and improve coordination. On the other, by structuring coordinating information, CRs provide the designer with a rich source of user and context information that can be leveraged by an adaptive support system at runtime.

In our methodology, coordination problems become opportunities for knowledge acquisition. An analytical procedure has been provided to identify these problems, and it can be applied to any media that a community of actors uses to coordinate a recurring activity. Relatively complete usage data is required to perform the prescribed analysis procedure. In the case of groupware that mediates the majority of the users' interactions, this may be achieved through the addition of logging utilities to the groupware system itself. For collaborations, which extend outside of the media under investigation, more traditional forms of data collection (e.g. videotape) will be required.

We have validated our approach via case studies that demonstrate its effectiveness in VesselWorld. The coordination problems experienced by VesselWorld users are highly representative of those problems in other collaborative environments. Users have different information about the world and need to share this information to accomplish their joint task. Individuals change objects in the world and need to inform others of these changes. Users have different roles and responsibilities. The task requires close coordination at times, but long-term commitments are also a critical part of the activity.

In this article, we've focused primarily upon problems related to grounding, but our analysis has revealed other types of coordination problems as well. Alterman et al. (2001) document the problems users faced in sequencing closely coordinated activities and tracking long-term commitments, and describe CRs that were developed to address these problems.

Similar kinds of problems have been described in the context of modern military applications. As networked technology has become ubiquitous in the military, electronic chat has found its way into mission-critical applications. Usage patterns reminiscent of the coordination problems in VesselWorld have been documented. Chat history is heavily relied upon to recall missed or forgotten information, and there are problems keeping track of mission orders (Heacox et al. 2004). The need for adaptive chat mechanisms that are more aware of the user's task so as to minimize interruptions (Cummings 2004) has also been noted.

Our approach is agnostic with regards to the particular inference technique that is used. Bayesian Networks were employed because they are easy to use, were sufficiently powerful for our needs, and many open-source and off-the-shelf implementations are available. Other techniques might have been used instead of or in addition to BNs. Sophisticated NLP algorithms could likely derive more mileage from references to wastes in chat than we were able to with our procedure. The training algorithms used to analyze the BN's performance could be employed at runtime to adapt to user behavior over time. More sophisticated planning algorithms and optimization routines could have been employed to produce efficient plans given available waste information. The point is that a range of possibilities for building adaptive support cascade from the increased availability of structured information about the user. The approach is then limited by the relevance of the information to the adaptation goal, the power of available inference algorithms and expertise of the designer.

The end result of applying our methodology to VesselWorld was a successful adaptive plan-generation component. This component is entirely domain specific, and may not be necessary or feasible in other domains. A natural extension to the existing methodology would offer guidance to the designer regarding the type of adaptive support that may be desirable or feasible using coordinating information from CRs. The development of such an extension is our current and future focus.

An avenue we are actively exploring is the generation of awareness information that is generally not available in collaborative applications. Much work on awareness has focused upon replicating information that is lost in distributed settings, such as awareness of what is going on (Dourish and Belotti 1992), who is around (Dourish and Bly 1992), and what has changed in a shared environment (Gutwin and Greenberg 1998). Recently, Carroll et al. (2003) has investigated the potential role of "activity awareness" in collaborative applications. The term activity awareness describes awareness of how work is embedded within the context of the overall activity. Coordinating representations are useful for generating the information required to provide this kind of awareness because they capture and structure information that constitutes the users' shared context. The computer, as a mediating artifact with significant abilities to summarize, sort, and synthesize structured data is in a good position to automatically combine and provide this information as feedback to users.

Acknowledgements This research was supported by the Office of Naval Research under grants No. N00014-96-1-0440 and N66001-00-1-8965. Additional support came from NSF grant EIA-0082393. The authors would also like to express their gratitude to the reviewers for their time and helpful insights.

Appendix A: Using chat

Users in VesselWorld often refer to wastes in chat using the labels they've assigned to objects in the Object List. To incorporate this information into the intent inference procedure, we modified the BN based on the following analysis.

Table 4 depicts the likelihood that a reference for an object will appear during the three consecutive 5 min windows prior to a lift of that object at time *t*. In the table, "Joint" and "Single" refer to whether a waste requires both or just one crane operator

Table 4 Probability of reference preceding a lift at time *t*

	t-5 to *t*		*t-10* to *t-5*		*t-15 to t-10*	
	Joint	Single	Joint	Single	Joint	Single
Lift	0.62	0.42	0.27	0.15	0.25	0.08
∼Lift	0.15	0.11	0.10	0.07	0.08	0.04

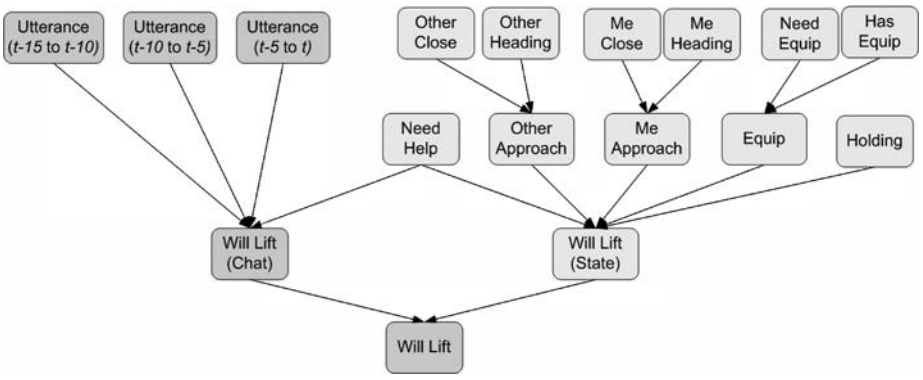

Fig. 15 BN with nodes for chat added; added nodes are colored differently

to lift. In the ∼*Lift* conditions, values reflect the likelihood some waste is referred to prior to a lift of a different waste.

There is about a 60% chance that waste will be referred to in chat in the 5 min preceding the lift if that waste requires assistance, and about a 40% chance if that waste can be lifted singly. Chat references are predictive of lift actions for roughly a 15-min window of time preceding a lift. On the basis of this analysis, we expanded our BN to include three 5 min windows of chat history, with one node for each 5 min window. The expanded network is shown in Fig. 15.

References

Albrecht, D., Zukerman, I., Nicholson, A.: Bayesian models for Keyhole Plan Recognition in an adventure game. User Model User-Adap. **8**(1–2), 5–47 (1998)

Alterman, R.: Rethinking autonomy. Mind. Mach. **10**(1), 15–30 (2000)

Alterman, R., Feinman, A., Landsman, S., Introne, J.: Coordinating representations in computer-mediated joint activities. In: Proceedings of the 23rd Annual Conference of the Cognitive Science Society, pp. 43–48. Boston, MA (2001)

Anderson, J.R., Corbett, A.T., Koedinger, K.R., Pelletier, R.: Cognitive tutors: lessons learned. J. Learning Sci. **4**(2), 167–207 (1995)

Bosma, W., Andre, E.: Exploiting emotions to disambiguate dialog acts. In: Proceedings of IUI'04, pp. 85–92. Funchal, Madeira, Portugal (2004)

Bauer, E., Koller, D., Singer, Y.: Update rules for parameter estimation in Bayesian networks. In: Proceedings 13th Annual Conference on Uncertainty in Artificial Intelligence (UAI)., pp. 3–13 (1997)

Brennan, S.: The grounding problem in conversations with and through computers. In: Fussell, S., Kreuz, R. (eds.) Social and cognitive psychological approaches to interpersonal communication pp. 201–225. Lawrenence Erlbaum, Hillsdale, NJ (1998)

Cañas A.J., Carvalho, M.: Concept maps and AI: an unlikely marriage? In: Proceedings of SBIE 2004: Simpósio Brasileiro de Informática na Educação, Manaus, Brasil (2004)

Carroll, J., Neale, D., Isenhour, P., Rosson, M.B., McCrickard, D.S.: Notification and awareness: synchronizing task-oriented collaborative activity. Int. J. Human-Comput Stud. **58**, 605–632 (2003)

Charniak, E., Goldman, R.: A Bayesian model of plan recognition. Artif. Intell. **64**, 53–79 (1993)

Clark, H.H., Brennan, S.A.: Grounding in communication. In: Resnick, L.B., Levine, J.M., Teasley, S.D. (eds.) Perspectives on socially shared cognition, pp.127–149. APA Books, Washington (1991)

Clark, H., Wilkes-Gibbs, D.: Referring as a collaborative process. Cognition, **22**, 1–39 (1986)

Clark, H.: Using language. Cambridge University Press, Cambridge, Great Britain (1996)

Cummings, M.: The need for command and control instant message adaptive interfaces: lessons learned from tactical tomahawk human-in-the-loop simulations. CyberPsychol. Behav. **7**(6), 656–661 (2004)

Dourish, P., Bellotti, V.: Awareness and coordination in shared workspaces. Proceedings of CSCW, pp. 107—114. Toronto (1992)

Dourish, P., Bly, S.: Portholes: supporting awareness in a distributed work group. Proceedings of the ACM CHI '92 Conference on Human Factors in Computing Systems, pp. 541–547 (1992)

Engeström, Y.: Activity theory as a framework for analyzing and redesigning work. Ergonomics, **43**(7), 960–974 (2000)

Feinman, A., Alterman, R.: Discourse analysis techniques for modeling group interaction. In: Brusilovsky, P., Corbett, A., de Rosis, F. (eds.) Proceedings of the Ninth International Conference on User Modeling, pp. 228–237 (2003)

Feinman, A.: From discourse analysis to groupware design. PhD Thesis, Computer Science Department, Brandeis University (2006)

Ferguson, G., Allen, J., Miller, B.: Trains-95: towards a mixed initiative planning assistant. In: Proceedings of the Third International Conference on Artificial Intelligence Planning Systems, pp. 70–77 (1996)

Ferguson, G., Allen, J.: TRIPS: an intelligent integrated problem-solving assistant. In: Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98), pp. 567–573. Madison, WI (1998)

Fischer, G.: User modeling in human-computer interaction. User Model. User-Adap. **11**(1–2), 65–86 (2001)
Goldberg, D., Nichols, D., Oki, B., Terry, D.: Using collaborative filtering to weave an information tapestry. Commun ACM. **35**(12), 61–70 (1992)
Goodwin, C., Goodwin, M.: Formulating planes: seeing as a situated activity. In: Middleton, D., Engeström, Y. (eds.) Cognition and communication at work, pp. 61–95. Cambridge University Press (1996)
Gutwin, C., Greenberg, S.: A descriptive framework of workspace awareness for real-time groupware. J. Comput. Support. Cooperat. Work. **11**, 411–446 (2002)
Heacox, N., Moore, R., Morrison, J., Yturralde, R.: Real-time online communications: 'Chat' user in navy operations. In: Proceedings of Command and Control Research and Technology Symposium. San Diego, CA. (Available online at: http://www.dodccrp.org/events/2004/CCRTS_San_Diego/CD/papers/086.pdf) (2004)
Hefley, W., Murray, D.: Intelligent user interfaces. International Workshop on Intelligent User Interfaces. Orlando, FL, pp. 3–10 (1993)
Horvitz, E., Breese, J., Heckerman, D., Hovel, D., Rommelse, D.: The Lumiere Project: Bayesian user modeling for inferring the goals and needs of software users. In: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, pp. 256–265. Madison, WI (1998)
Horvitz, E.: Principles of mixed-initiative user interfaces. In: Proceedings of CHI' 99. pp. 159–166. Pittsburgh, PA (1999)
Hutchins, E.: Cognition in the wild. MIT Press, Cambridge, MA (1995)
Jameson, A.: Adaptive interfaces and agents. In: Jacko, J.A., Sears, A. (eds.) Human computer interaction handbook, pp. 305–330 Erlbaum, Mahwah, NJ (2002)
Jameson, A., Baldes, S., Kleinbauer, T.: Generative models of group members as support for group collaboration. Workshop on User and Group Models for Web-Based Adaptive Collaborative Environments: Proceedings of the 9th International Conference on User Modeling, pp. 1–14 (2003)
Kaiser, E.: Multimodal new vocabulary recognition through speech and handwriting in a whiteboard scheduling application. In: Proceedings of IUI'05, pp. 51–58. San Diego, CA (2005)
Landsman, S., Alterman, R., Feinman, A., Introne, J.: Vesslworld and ADAPTIVE, Brandeis University Tech Report CS-01-213; Presented as a demonstration at Computer Supported Cooperative Work, 2000 (2001)
Landsman, S., Alterman, R.: Using transcription and replay in analysis of groupware applications. Brandeis University Technical Report CS-05-259 (2005)
Lau, T., Domingos, P., Weld, D.: Version space algebra and its application to programming by demonstration. In: Proceedings of the Seventeenth Int'l. Conf. on Machine Learning, pp. 527–534 (2000)
Leong L.H., Kobayashi, S., Kshizuka, N., Sakamura, K.: CASIS: A Context Aware Speech Interface System. In: Proceedings of the 10th International Conference on Intelligent User Interfaces, pp. 231–238 (2005)
Lesh, N, Rich, C., Sidner, C.L.: Using plan recognition in human-computer collaboration. In: Proc. 7th Int. Conf. on User Modeling, pp. 23–32 (1999)
Maes, P.: Agents that reduce work and information overload. Commun. ACM. **37**(7), 30–40 (1994)
Malone, T., Lai, K., Grant, K.: Two design principles for collaboration technology: examples of semiformal systems and radical tailorability. In: Olson, G., Malone, T., Smith, J. Coordination Theory and Collaboration Technology, pp. 125–160. Lawrence Erlbaum Associates, NJ (2001)
McLaren, B., Walker, E., Harrer, A., Bollen, L., Sewall, J.: Creating cognitive tutors for collaborative learning: steps toward realization. In this issue (2006)
Miller, G. (ed.): Five papers on wordNet. Special Issue Int. J. Lexicography **3**(4) (1990)
Nardi, B.: A small matter of programming. The MIT Press, Cambridge, MA (1993)
Norman, DA.: Cognitive artifacts. In: Carroll, J.M. (ed.) Designing interaction: psychology at the human-computer interface, pp. 17–38. Cambridge University Press (1991)
Novak, J.D., Gowin, D.B.: Learning How to Learn. Cambridge University Press, New York, NY (1984)
Pan, S., Shen, S., Zhou, M., Houck, K.: Two-way adaptation for robust input interpretation in practical multimodal conversation systems. Proceedings of the 10th International Conference on Intelligent User Interfaces. pp. 35–42 (2005)
Pearl, J.: Probabilistic reasoning in intelligent systems. Morgan Kaufmann, San Francisco, Calif (1988)
Read, T., Barros, B., Bárcena, E., Pancorbo, J.: Coalescing individual and collaborative learning to model user linguistic competences. In this issue (2006)
Rich., C., Sidner, C.L.: COLLAGEN: a collaboration manager for software interface agents. User Model. User-Adap. **8**(3–4), 315–350 (1998)

St. Amant, R., Dinardo, M., Buckner, N.: Balancing Efficiency and Interpretability in an Interactive Statistical Assistant. In: Proceedings of IUI'03. (Miami, FL, 2003). ACM Press, New York, 2003. 181–188 (2003)
St. Amant, R., Young, R.: Interface agents in model world environments. AI Magazine **22**(4), 95–108 (2001)
Schmidt, K., Simone, C.: Coordination mechanisms: towards a conceptual foundation of CSCW systems design. Comput. Support. Cooper. Work. J. Collaborat. Comput. **5**(2–3), 155–200 (1996)
Schneiderman B.: Designing the user interface. Addison Wesley (1998)
Shneiderman, B., Maes, P.: Direct manipulation vs. interface agents: a debate. Interactions **4**(6), 42–61 (1997)
Sidner, C.L.: An artificial discourse language for collaborative negotiation. In: Proceedings of the Twelfth National Conference on Artificial Intelligence. AAAI Press, Menlo Park, CA., pp. 814–819 (1994)
Soller, A.: Computational modeling and analysis of knowledge sharing in collaborative distance learning. User Model. User-Adapt. **14**, 351–381 (2004)
Suchman, L., Trigg, R.: Understanding practice: video as a medium for reflection and design. In: Greenbaum, J., Kyng, M. (eds.) Design at Work: Cooperative Design of Computer Systems, pp. 65–90. Lawrence Erlbaum Associates, Hillsdale, New Jersey (1991)
Suebnukarn, S., Haddawy, P.: Modeling individual and collaborative problem-solving in medial problem-based learning. In this issue (2006)
Suthers, D.: Representational guidance for collaborative inquiry. In: Andriessen, J., Baker, M., Suthers, D. (eds.) Arguing to learn: confronting cognitions in computer-supported collaborative learning environments. pp. 27–46. Netherlands, Kluwer (2003)
Traum, D.: A computational theory of grounding in natural language conversation. Doctoral Thesis. Technical Report TR545. University of Rochester (1994)
Vincente, K.: Cognitive work analysis: toward safe, productive, and healthy computer-based work. Lawrence Erlbaum Associates, Inc New Jersey (1999)
Zuckerman, I., Litman, D.: Natural language processing and user modling: synergies and limitations. User Model. User-Adapt. **11**, 129–158 (2001)

Authors' Vitae

Joshua Introne is a Ph. D. candidate in Computer Science at Brandeis University. He received his A.B. in Computer Science and Mathematics from Bowdoin College in 1995, and his M.S. degree in Computer Science from Brandeis University. His primary interests lie in the design of adaptive systems, although his other research has included the development of decision support tools for military and security analysts for Charles River Analytics, Inc. This article summarizes the current state of his thesis work on building adaptive groupware.

Dr. R. Alterman is Professor of Computer Science at Brandeis University with a joint appointment in the Volen Center for Complex Systems. He received a B.A. degree in Computer Science from UC Berkeley and a Ph.D. degree in Computer Science at UT Austin. Dr. Alterman has worked in several areas of cognitive science and artificial intelligence, including planning, situated activity, distributed cognition, text, discourse, and case-based reasoning. He has done extensive work as a cognitive modeler, and more recently has investigated methods and techniques for cognitively engineering computer-mediated collaborative activity. He has authored over seventy technical papers.