Extracting Procedural Knowledge from a Groupware for Planning System

Joshua Introne, Rick Alterman

Volen Center for Complex Systems
MS018 Brandeis University
Waltham, MA 02454
[jintrone,alterman]@cs.brandeis.edu

Introduction

For the past three years, GROUP (Group for Research On Usage and Pragmatics) has been developing the VesselWorld system. In VesselWorld, three participants collaborate on remote workstations to remove barrels of toxic waste from a harbor. Each participant is the captain of a ship, and their joint goal is to remove all of the barrels from the harbor without spilling any toxic waste. Each ship has a geographically limited view of the harbor and some unique capabilities. To safely remove a barrel of waste from the harbor requires varying degrees of coordination between the participants. Figure 1 is a snapshot of the main window (the Control Center) in VesselWorld for one of the ship captains.

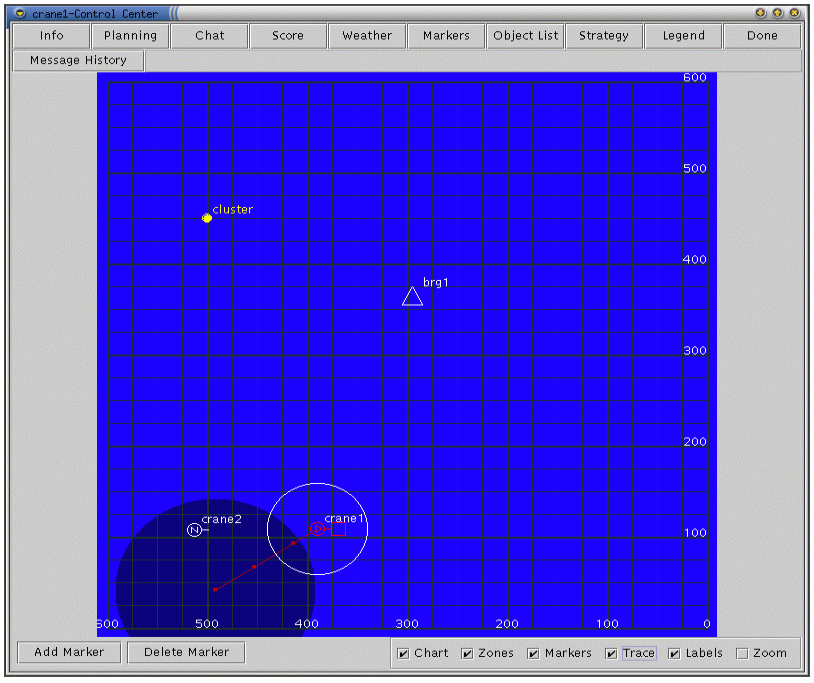

Figure 1: VesselWorld Control Center

Managing the VesselWorld interface is a complex and time-intensive task for the user. The system might ultimately reduce this burden by recognizing user plans and then automatically manipulating the interface to assist participants in executing their plans. To do this, the system must first acquire a library of shared and individual plans typically used by the group. This library may then be used to recognize and act upon the participants' joint behavior in future problem solving sessions. *In this paper we describe a technique for acquiring procedural knowledge that makes use of information users must exchange to stay coordinated in a joint task.* The key insight is that tools may be introduced into the system which users want to use, and that these tools both improve users' performance and significantly reduce the complexity of extracting procedural knowledge from usage data.

Staying Coordinated

Two versions of the VesselWorld system have been implemented. In the base system, users can only communicate via a chat window. The enhanced version of the system introduces three shared semi-structured representations for communicating information relevant to planning. In particular, the representations characterize users' high-level shared plans (Grosz and Sidner, 1990), make explicit shared references (Clark & Wilkes-Gibbs, 1990) to various objects in the domain, and simplify sequencing of coordination points (Alterman and Garland, in press).

To design the semi-structured representations, over eighty hours of data from several user groups using the base VesselWorld system was collected and analyzed. This analysis identified several recurring types of coordination breakdowns, and subsequent interviews with user groups verified this. Below is an example of usage data (only the information extracted from the chat window is shown here) that demonstrates such a coordination breakdown. In this dialogue two ship captains are doing a joint lift and carry, but do not closely coordinate on entry and exist phases, and so get out of synch and drop the barrel of toxic waste (line 7).

```
1. 10406910 crane2 : I'm at lw4. Any spec. equip?
2. 10460863 crane1 : nonenone
3. 10500140 crane2 : Ok I've deployed the crane and will attach at next move
4. 10518860 crane1 : ok, and lift
5. 10633890 crane2 : let's head over to the barge and dump her. let me know
6. when to dump
7. 10668900 crane2 : I think we dropped it
8. 10701610 crane1 : i know but i don't see you or the waste any more
9. 10762511 crane2 : Move along side it and we will try again
11. 10831340 crane1 : ready to lift?
12. 10853251 crane2 : not yet ihaven't deploy and i'm not close enough to you
13. 10892720 crane2 : I'll tell you when
14. 10965830 crane2 : Ok,c1
15. 11010104 crane2 : Hold on I can't find you
```

Figure 2: Coordination Failure

The three components which were developed to overcome coordination difficulties that were identified are the Strategic Planning window, the Object List, and the Shared Planning window. Each of these components is shared, so that the same information is seen by all users in these components at any given time.

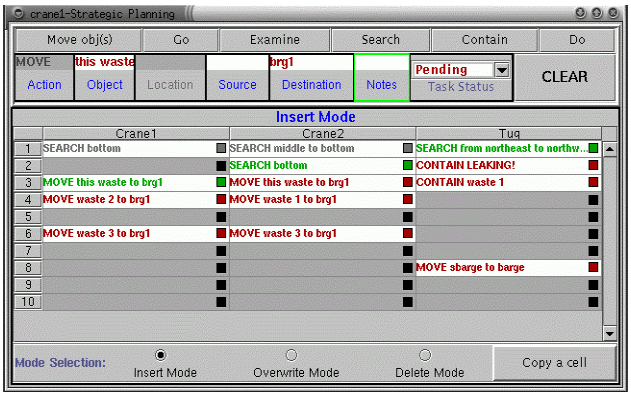

Figure 3: Strategic Planning

- **Strategic Planning**

  The Strategic Planning window allows participants to share information about high-level plans and goals. It consists of three columns side by side, one for each captain, to allow users to specify current and future intentions, and indicate relative positioning of these intentions. Entries in each column may be moved around, edited, and marked to indicate whether the status of the item is active, pending, or complete. Six high-level operators are provided with which to compose entries in the Strategic Planning window. These operators were chosen on the basis of chat traces from preliminary trials, and the close examination of one expert group. Each operator may be further specified by filling out supporting fields. These fields may be filled out by clicking on the world interface or object list, or typed in directly.

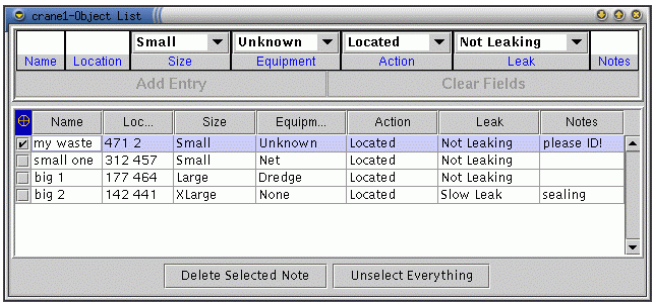

Figure 4: The Object List

- **The Object List**

  The Object List helps make references to objects in the world explicit. Each entry in this list contains several fields of information including an user assigned name, the status, and the location of the associated object. The location field may be filled by clicking on the field and then upon an object in the world. A free text field is also provided for each entry so that any other relevant information may be communicated. Icons representing objects with valid locations may be displayed in the Control Center (Figure 1) interface, to assist users in mapping items in the list to objects in the world.

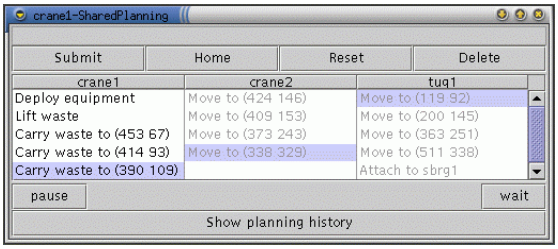

Figure 5: Shared Planning window

- **Shared Planning**

  The Shared Planning window is a view of the steps each actor has actually entered for future submission. This is a modification from the base system, in which a user may only see his or her own plans. The Shared Planning window provides a means for close coordination without explicit communication (for example, in the case where two users must lift a barrel at the same time), and can help to make low-level actions explicit where the strategy window does not provide the appropriate granularity of information.

Because the Object List and Strategy Component are semi-structured, the number of low-level (e.g. keystroke) actions users must perform to communicate information these components are designed to handle is reduced, while the ability to communicate information beyond what has been anticipated in their design is preserved (Malone et al., 1986). The structure of these components also serves as a reminder for participants of information necessary to stay coordinated. Experiments are currently underway which will confirm that the semi-structured representations improve the participants' performance (as determined by several criteria) over chatting alone.

Segmenting the data

All user activity is logged by the VesselWorld system. The task for the system is to segment this usage data into goal-oriented procedural representations. This is a very difficult task in the base system for several reasons: interface actions pertaining to different goals are interleaved, both for the group and for the individual; interface actions themselves are not marked in any way to indicate which goal, if any, an action serves; references to objects in the chat transcripts vary in format, rely upon contextual information, and contain frequent mistakes; users do not announce intentions and plans in the chat traces in any canonical fashion, and so this information must be extracted by hand or an NLP system.

The data provided by the semi-structured representations significantly reduces the complexity of the segmentation task. The Strategy Component partially segments the data into regions of goal-directed behavior, providing a hint as to what goal user actions correspond to. The Object List establishes canonical references within the system, and moves much of the object-specific information from unstructured chat to a structured format which is readily interpreted by the system. This additional information helps provide the system with the context necessary to reliably tease apart many interleaved interface actions into threads of goal directed activity.

An Example with the Object List

The first step in building a procedural model of user activity is to sort the usage data into threads of activity which refer to objects. In the base VesselWorld system, the only data available to the system for determining what interface actions refer to which objects are chat traces and individual plans. Users may refer to objects using a variety of methods, and discovering these is a difficult task for the system. With the Object List representation, users tell both the system and other participants what names reference which objects. Subsequent communication regarding these objects via chat, the Strategic Planning window, or Object List itself, usually makes use of these references, and the object referred to in such communication can thus be resolved by the system with minimal effort.

One step in creating entries in the Object List is clicking on a field in the Object List and then clicking on an object in the World View. The action of clicking on an item in the World View generates an event which logs the connection between the object and a unique identifier used internally by the system. Users create a name for the object in a free text field, and update other fields appropriately. Status information regarding an object is frequently passed through the supporting fields in an Object List entry rather than exchanged in chat. Communication about the object in chat will very likely use the name established in the Object List. A thread of activity referring to an object in the world may be identified by reading the usage data with a modified state machine that collects events which are likely to refer to a given waste. Figure 6 is a simplified version of portion of such a state machine. The circled numbers in the diagram indicate the steps in setting up a container for the history of activities associated with an object. These steps are as follows:

1. The user enters info mode, which is necessary to get full information about an object, and clicks on an object in the interface to retrieve that information. The generated event contains the system's internal name for the waste (not available to the user) and a location. If no existing waste with this name has yet been discovered, a candidate waste history container is generated.
2. The user clicks on a field in the Object List, and then clicks on the world view to instantiate the Location field. If this location matches (or is close enough to) a location in a candidate container for an object history, the candidate is marked as resolved by the system.
3. Additional information is added to the resolved object history container as that user, or other users, modify fields in that Object List entry.

Nearly all subsequent user-generated events concerning this object may be identified using the established references and can be collected into the history container.

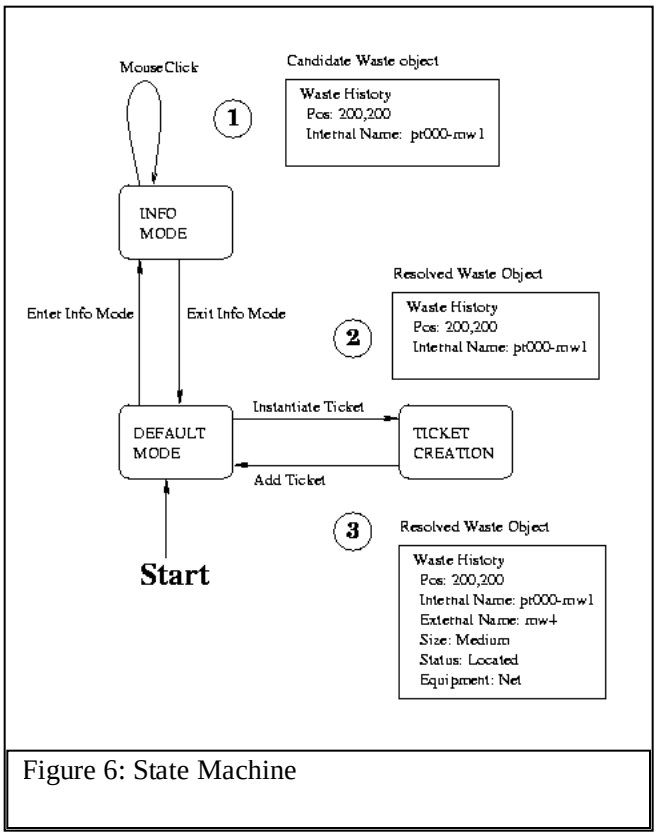

Figure 6: State Machine

A object history that has been generated by such a state machine is shown in Figure 7.

Building Procedural Knowledge

After many such activity threads have been extracted, they will be used to develop a case base for goals in the VesselWorld environment. The Multicase model (Zito-Wolf) is the current best candidate for the case base architecture. In Multicase, multiple episodes are represented as a directed network of decision points, which can enumerate both encountered execution paths and explicit instructions to achieve a goal. Multicase is ideal for the task at hand, because it compiles all prior experience directly into a procedural representation. Several modifications, however, will need to be made to Multicase to tailor it to group activity.

Experiment

The research group is currently collecting data with the enhanced version of the VesselWorld system. Approximately forty hours of usage data have been collected thus far, with a target of over one hundred hours of data with this system. Half of this data will be used to train a predictive model in the manner described above, and the effectiveness of the model will be tested against the remaining data.

Summary and Future Direction

Collaboration and joint activity require awareness of other actors' goals, plans, commitments, and intentions (Grosz & Kraus, 1996; Bratman, 1999; Levesque, Cohen, and Nunes,

```
pr000-mwaste1
*************
4364     crane2  pr000-mwaste1 java.awt.Point[x=427,y=55,java.awt.Point[x=183,y=60]
4508     crane2  pr000-mwaste1 {Location=183 60, Size=Medium, Action=Located, Equipment=Unknown, Leak=Not Leaking, Notes=,
         Name=mw4}*
4952     tug1 pr000-mwaste1 java.awt.Point[x=428,y=548],
5095     tug1 pr000-mwaste1 {Location=183 60, Size=Medium, Action=Located,  Equipment=None, Leak=Not Leaking, Notes=, Name=mw4}
11719    crane1  pr000-mwaste1 I'm going to go do mw4 while we're waiting.
11818    crane1  pr000-mwaste1 {Notes=, Source=, Action=DO, Destination=, Object=mw4,Color=java.awt.Color[r=178,g=0,b=0],
         Location=,Agent=Crane1}
13128    crane1  pr000-mwaste1 {Notes=, Source=, Action=DO, Destination=, Object=mw4, Color=java.awt.Color[r=0,g=178,b=0],
         Location=, Agent=Crane1}
14151    crane1  pr000-mwaste1 [(LIFT [crane1] {} pr000-mwaste1), (CARRY [crane1] {288 76}), (CARRY[crane1] {334 86}), (CARRY
         [crane1] {380 96}), (CARRY [crane1] {426 106}), (CARRY [crane1] {472 116}), (CARRY [crane1] {517 126}), (LOAD [crane1] {}
         barge1)]
14468    crane1  pr000-mwaste1 {Location=ih, Size=Medium, Action=Located, Equipment=None, Leak=Not Leaking, Notes=, Name=mw4}
15589    crane1  pr000-mwaste1 I'm heading to the brg with mw4.
17955    crane1  pr000-mwaste1 {Notes=, Source=, Action=DO, Destination=, Object=mw4, Color=java.awt.Color[r=128,g=128,b=128],
         Location=, Agent=Crane1}

18171    rane1   pr000-mwaste1{Location=brg, Size=Medium, Action=Located, Equipment=None, Leak=Not Leaking, Notes=, Name=mw4}
```

Figure 7: Waste History

1990), so that effective coordination may take place. In collaborative tasks where there is a separation of time or space or both, coordination is more difficult because intentions, plans, goals, references, and coordination

points must be explicitly communicated. Semi-structured representations facilitate such communication, and, if properly designed, users will want to use them. These components also reduce the guesswork in determining what the users' intentions because the structured communication is more easily interpreted by the system.

Three interface components have been discussed in this abstract which make the task of communicating plans and intentions easier, and structure the information in a useful fashion. Experiments with these new components are underway, and we will soon be able to quantitatively describe to what degree these components actually assist users, and if the information they provide though structure is sufficient to build procedural knowledge.

By storing goal-based procedural knowledge in a case base it will be possible to create a system which reduces the amount of work users must do to accomplish their goals in the task domain. Case retrieval and predictions provided by our representation can be used to support automation of interface actions, such as moving and resizing windows, or highlighting of important information. Cases might ultimately be retrieved to offload some of the work in planning. Tracking breakdowns in coordination with respect to user intentions will indicate areas where procedural knowledge may be reintroduced to assist novices. Because usage data has been obtained in conjunction with a reliable representation of user intentions and goals, reconstruction of domain specific procedural knowledge and the effective reapplication of this knowledge to assist users will be possible.

References

Alterman, R. and Garland, A. (2000) Convention in joint activity. *Cognitive Science*, To Appear.

Alterman, R. (2000) Rethinking Autonomy. *Minds and Machines*, 10:1 15-30.

Alterman, R., Landsman, S., Feinman, A., Introne, J., and Kirchenbaum, S. (1998) *Groupware for Planning*, Brandeis University Computer Science Department, Technical Report CS-98-200.

Bratman, M. E. (1999) Shared cooperative activity. In *Faces of Intention*, pp. 93-108. Cambridge University Press, 1999.

Clark, H. H. and Wilkes-Gibbs, D. (1990). Referring as a collaborative process. *Cognition*, 22:1-39.

Grosz, B. and Kraus, S. (1996). Collaborative plans for complex group action. *Artificial Intelligence*, 86:269-357.

Grosz, B. and Sidner, C. (1990). Plans for discourse. In Cohen, P. R., Morgan, J., and Pollack, M. E., editors, *Intentions in Communication*, pp. 417-444. Bradford Books, Cambridge, MA.

Levesque, H. J., Cohen, P. R., and Nunes, J. H. T. (1990). On acting together. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pp. 94-99.

Malone, T. W., Grant, K. R., Lai, K., Rao, R., Rosenblitt, D. (1986) In *Proceeding of the Conference of Computer-Supported Coorperative Work.*

Zito-Wolf, R. J., (1993). *Case-Based Representations for Procedural Knowledge*. Ph.D. thesis, Brandeis University.