A PARTICIPANT CONSTRUCT SYSTEM

Mildred L. G. Shaw & Ernest Chang
Alberta Research Council, 6815 8th Street NE, Calgary, Alberta, Canada T2E 7H7
& Department of Computer Science, University of Calgary, Alberta, Canada T2N 1N4

Advances in computer technology make it feasible to develop systems for group problem solving, participant systems in which a number of people interact with one another through personal computers on a network. This paper introduces the notion of a *participant system* and gives examples of such systems for teleconferencing and personal construct psychology. The design considerations and detailed examples of a *participant construct system* using interactive repertory grid elicitation and exchange grid methodologies are described for a network of Macintosh computers.

Introduction

This paper describes a system for interactively eliciting, exchanging and comparing repertory grids from multiple simultaneous users on a network of personal computers. The *participant construct system* (PCS) which does this has been designed as an aid to group problem-solving (Shaw 1986a,b). The users of PCS are attempting to extend and understand their own thinking and problem-solving capabilities by interacting with other people. They are able to:

- see the relationship of their points of view to those of others;
- explore differing terminology for the same constructs;
- become aware of differing constructs having the same terminology;
- extend their own construct systems with those of others;
- provide others with constructs they have found valuable;
- explore a problem-solving domain using the full group resources.

The manual elicitation of construct systems using structured interview techniques is tedious, time-consuming, and requires skill, practice and an appropriate personality. In recent years computer-based techniques have been developed for on-line elicitation and analysis of construct systems. These have proved very successful and are now in widespread use (Shaw 1980, 1981a). One of the most interesting developments of such systems has been to attempt to combine the construct systems of a number of people, make them accessible to all and hence share their knowledge and skills (Shaw 1981b, 1985, Shaw & Gaines 1981, Boose 1986). The extension of interactive construct elicitation techniques to groups exemplifies the operation of a *participant system* (PS)—computer-based communication between multiple users that is essential to the performance of their tasks, rather than merely incidental to their use of a timeshared computer or computer network (Chang 1986a,b).

Participant Systems

The concept of a participant system is best presented by contrasting it with the rationale behind the majority of current multi-user computer systems. Interactive timeshared computers were developed in the 1960s to give each user a virtual machine that emulated personal access by that user to a dedicated computer. The design requirement was that the interaction of other users with the time-shared computer should be as far as possible not apparent. Timesharing could never fully achieve this objective since CPU availability and disk access were subject to the fluctuating demands of other users and this could be strongly apparent to a user as varying time delays that were disconcerting because they had no relation to the tasks he had initiated.

The decline in cost of computer hardware in the 1970s made it possible to provide personal computing through dedicated actual machines rather than timesharing virtual machines. Personal computers satisfy the original design requirements for timesharing systems and the only reasons, within those requirements, for a user to access another system are: access to greater computing power such as an array processor; access to greater information storage such as high-volume files; access to information updated remotely such as a centralized database.

It is interesting to note that the first two requirements, for computing power and storage, are being increasingly satisfied by personal computers. Only the third, access to remotely updated information, has a fundamental, systemic component requiring a communication capability. It is this third requirement that the concept of participant systems addresses—the interaction of multiple users as an essential requirement of their individual, or mutual, tasks.

Fig.1 shows a typical configuration for a participant system implemented as a network of personal computers. The group of persons, who have come together to participate in the system, each access it through a personal computer acting as a terminal to a distributed computing network. Terminals to a timeshared central computer could provide an alternative implementation but the delay-free interaction of personal computers is preferable in giving a highly responsive system, particularly for graphic interaction. The database shown provides storage for information that may be shared among the participants. They may communicate with the database, or one another, through the network.

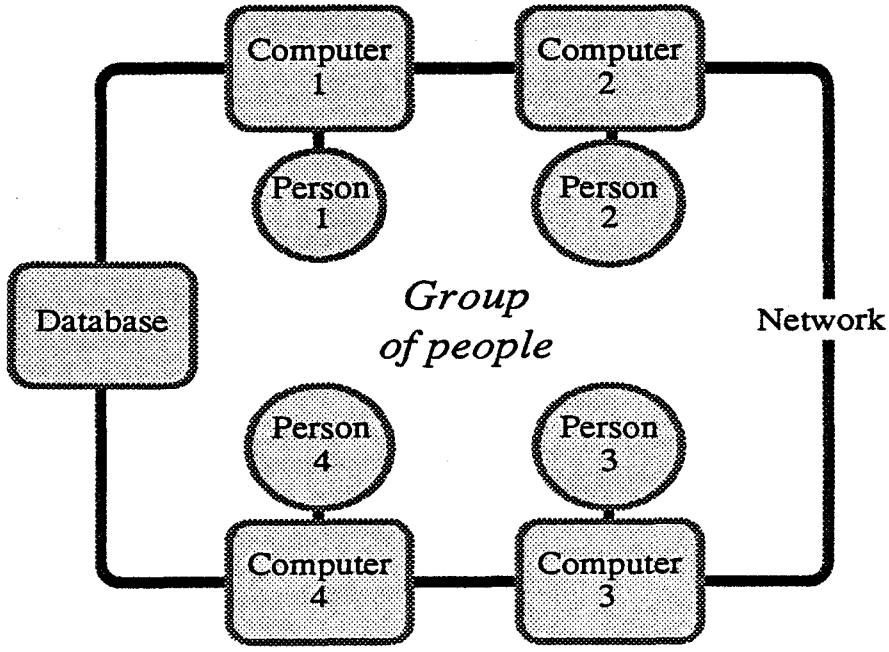

Fig.1 Typical participant system implementation

Note the emphasis on a group of people—the diagram could be of individuals, each carrying out separate activities on a computer network. However, it is the essence of a PS that those using it have some joint purpose, giving them coherence as a group.

CANTATA: A Teleconferencing Participant Systems

Cantata (Chang 1986a) is a PS for conferencing that was originally implemented at the University of Victoria on a network of IBM-XT's linked through Ethernet, and is being implemented at the ARC on Macintoshes networked through Appletalk (Kasperski, Mellen

& Chang 1986). It broadcasts text messages from one user to all others on the network, displaying them in windows that are specific to the sender. The windows are sizable, scrollable, selectably visible or invisible, and support word-wrap. A status message is displayed when text comes to an invisible window. Apart from this, no user interaction protocols are implemented within Cantata itself. Communication can exist at task and meta-task levels, but since both map on to text messages, no explicit mechanism for meta-task communication has been created as yet. In Cantata, there are no explicit rules for communication, since everyone can address everone else at the same time for as long as each likes with no specific topicality; concepts like turn-taking, topic switching, decision-making, must be dealt with through structures that evolve from the participants themselves. In these ways, Cantata is different from most computer conferencing systems, which by contrast, suppport asynchronous communications and a bulletin-board model.

This lack of enforced interaction protocol gives Cantata users great flexibility and freedom, but also creates the problem of coping with a novel communication situation and developing protocols to operate effectively. Cantata users face an enigmatic situation. On the one hand they are able to receive messages as they are constructed, and can respond to one or more messages before the sender(s) complete them. The ability of the eye to read much faster than the hand can type means that the recipients can understand and act upon several input streams in parallel, while messages are being constructed. This is therefore a medium that has no analog in the normal world of personal interaction: you can interrupt without interrupting, and listen to three people at the same time in Cantata. On the other hand, the absence of structure in the communications among participants causes some confusion, partly because of the newness of this medium. It is difficult to get away from the usual cultural expectation that only one person in a conversation has the floor, that there are leadership or consensus methods for changing or maintaining topics, that feedback is expected as one speaks. It is difficult, at present, to know whether the advantages of this kind of medium can be better exploited by building structuring mechanisms into the system, or by more meta-conversational exchanges, so that participants can create rules for using Cantata dynamically, and follow them voluntarily.

The design, development and application of participant systems requires an understanding of both the computer technology involved and the psycho-socio-dynamics of the users. The participant construct system design addresses this requirement by designing a specialist PS for a domain where large-scale cognitive psychological and sociological theories already exist, implementing it on a network of Apple Macintosh personal computers where an appropriate human-computer interface already exists. The highly structured knowledge elicitation techniques developed for the PLANET system (Shaw 1982, Shaw & Gaines 1986) provide a working environment for the users of PCS which avoids the problems of direct, unstructured interaction in Cantata. Ultimately, both informal and structured interaction must be combined to allow the full capabilities of PSs for group problem solving to be exploited.

Personal Construct Psychology

Kelly's (1955) Personal Construct Psychology (PCP) provides a systemic model of the cognitive and behavioral processes of the individual, including inter-personal and social interaction. A person is modeled as *construing* her experience by dividing it up into chunks (elements) and classifying these using a system of bipolar constructs. Constructs are themselves construed by other constructs and hence form a hierarchical system. Construing is used to enable a person to anticipate the future as a hypothesized replication of the past. Constructs are changed as a result of errors in anticipation so that personal philosophies are seen as essentially fallibilist. A person will be more willing to change some constructs rather than other, core, constructs, giving a further order relation on the construct system.

In PCP, learning is seen as the development of the construct system, social processes are seen as the development of shared constructs, and problem-solving is seen as the active anticipation of the future. The theory provides complete foundations for all human psychological and social processes. Hence it is possible to build technologies based upon it which are very general in concept and application. For example, an expert system in PCP terms is a computer-based system in which the relevant construction system of an expert has been encoded in an active form (Shaw & Gaines 1983).

In computational terms the elements in PCP may be seen as entities, the constructs as attributes and the relation between elements and entities as values in an entity-attribute database. Kelly terms such a database a repertory grid and Fig.2 shows how a typical grid may be represented. The elements in this grid are people represented by their initials. The three constructs illustrate the mix between subjective and objective possible in personal constructs.

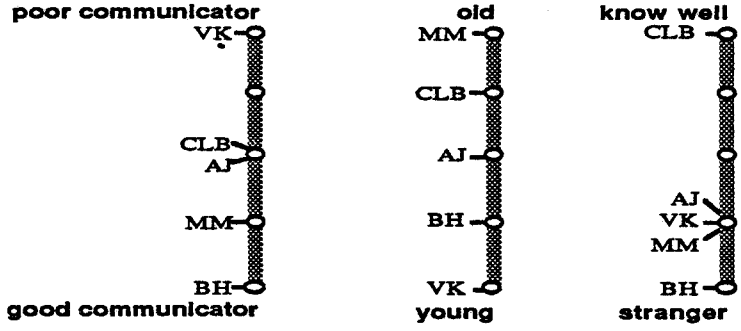

Fig.2 A repertory grid of people on three constructs

Participant Construct System Design

With the previous discusion in mind, and based on past experience with the PLANET system, the PEGASUS individual grid elicitation procedure was taken as an initial model for PCS. Interaction with PEGASUS, with feedback of construct and element matches, is known to be a stimulating and enlightening experience. Then to ensure that the group interaction through PCS is seen as important in its own right, the features have been included to:

- Select additional elements from those entered by other participants;
- Compare one's own constructs with those of other participants;
- Exchange complete grids with other participants and evaluate mutual agreement and understanding.

Such *exchange grid* procedures are known to aid individuals in coming to understand their agreements and disagreements with the way others perceive the world, and to help them to understand and use the constructs of others (Shaw 1980). They give a basis for group problem solving and hence PCS may be presented to interested users as a participant system around which group problem solving procedures may be built.

In choosing a technology for the implementation of PCS the importance of a simple and attractive user interface has been paramount. PEGASUS was designed as a structured interview for users with little or no knowledge of problem solving techniques. The program largely controls the interaction and limits the user's choices to a few significant decisions. Greater flexible was both desirable in PCS and also intrinsically required by the overlay of group interaction on the basic construct elicitation procedures.

A poor person-computer interface can completely negate the value of the most advanced participant system. This has been a major consideration in the PCS design and the Apple Macintosh was chosen as a user terminal to PCS because:

- The windowing facilities make it possible to present multiple channels of information in a modular and comprehensible fashion.
- The graphic data presentation makes it possible to present grid data structures in an attractive and comprehensible format, replacing the display of numeric ratings with one based on spatial relations.
- Much of the interaction with PCS can consist of spatial manipulation using the mouse to click and drag objects on the screen, rather than typing text or commands.
- The proper use of the toolbox routines leads to a uniform interface that is consistent with other Macintosh programs that users already know.
- The Macintosh has a reputation for being simple to understand and easy to use without training or reading manuals, reducing user anxiety and encouraging expectations of being able to operate the system.

The Macintosh interface encourages an object-oriented programming paradigm in which the user has access to a number of discrete, manipulable objects, most of which are potentially accessible at any time. That is, there is very little imposed flow of control. This gives the system tremendous flexibility, allowing much freedom in use, and enabling the users to impose their own mode of working upon the system rather than having to conform to some preset sequence. However, flexibility and freedom are not in themselves necessarily advantageous in human-computer interaction. Users may find the decision-making necessary to drive the system too great a burden, or they may never discover how to use the system effectively because it gives little indication of reasonable modes of use. Thus the Macintosh interface in its own right is no panacea for effective human-computer interaction. It needs careful system design based on an understanding of its underlying concepts, functionality and the psychology of interaction.

These considerations led to the following design rules for the PCS user interface on the Macintosh computer:

- Follow the Macintosh standards (defined by the toolbox and its use by programs such as the Finder, MacWrite and MacDraw) as closely as possible so that Macintosh users find PCS natural in use.
- Make each object defined by a window a natural concept unit of the grid elicitation and analysis process so that its purpose and functionality are obvious.
- Design each object to be self-describing so that minimal explanation and use of the manual are necessary.
- Use a graphic representation of data structures and results of analysis.
- Base user data entry on object manipulation using the mouse wherever possible.
- Make the system tolerant of user actions giving ease of system exploration through the capability to reverse actions and recover from errors.
- Within these constraints offer the user the maximum freedom of action possible.

These rules are largely ones of Macintosh programming style that any program should follow. There are, however, many commercial programs that deviate from this style, often because they have been transfered from other computers. Worse, there are a number of programs also that apparently conform to the style but fail to follow it in some important features. These latter programs undermine the user's confidence in the system and his capability to understand it and it has been important to avoid such problems in PCS.

PCS in Operation

The illustrations of PCS in this section are taken from the *PCS User Manual* (Shaw 1986a). In this example Bill is construing *managers* to see which of the ones he knows are *effective* and why. When starting a new topic the status window must first be checked and changed if necessary. Usually a new topic will have a specific purpose which should be entered in the appropriate box, and kept in mind as the interaction proceeds. The bottom half of the window will change automatically as elements and constructs are added (Fig.3).

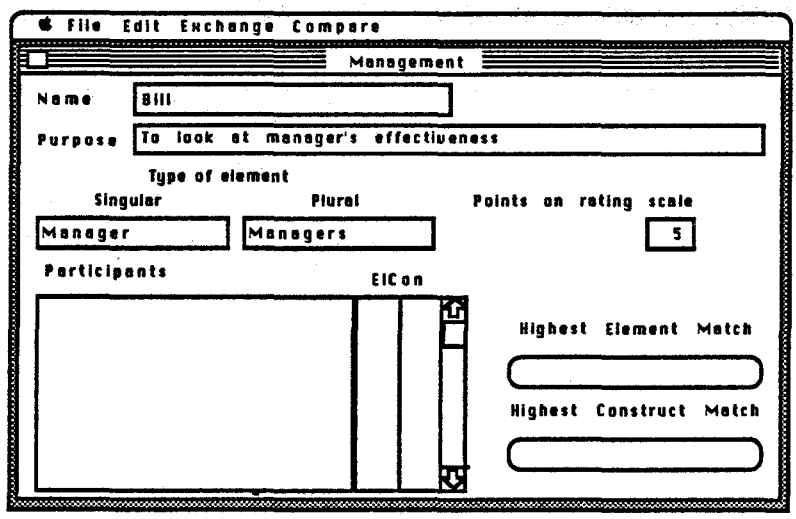

Fig.3 PCS status window

The next thing for Bill to do is to think of some managers and add these as elements. He does this by going to the *Elements* window in the *Edit* pull-down menu, then typing in the names of the managers he would like to think about one on each line. He adds five (Fig.4).

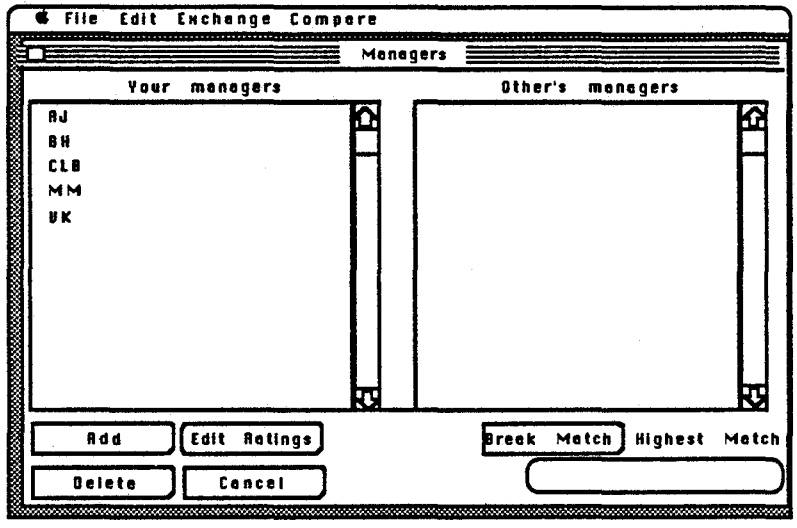

Fig.4 PCS elements window

As Bill is just starting his grid, he decides to elicit a construct from a triad, so he goes to the *Constructs* window in the *Edit* pull-down menu, and chooses triadic elicitation from a random triad. Sometimes part way through the elicitation the user may wish to choose his own triad, otherwise the system will choose one at random. Bill sees the three elements in the triad, and decides that VK is the one which is different from AJ and BH (Fig.5).

Next Bill has to decide why he thinks AJ and BH are alike and different from VK. He types in the description of the pair, *don't need supervision*, on one pole of the construct (Fig.6). Then he names the other end—that is, the way in which VK is different is that he *needs supervision*. This defines Bill's scale from *don't need supervision*, to *needs supervision*. He now has to rate all his other elements on this scale. If he finds that his original triad AJ, BH and VK need to be moved to accommodate the others he can do that. In this case he puts CLB partly up the scale, and MM partly down the scale, because MM doesn't need much supervision and CLB needs a bit more but not as much as VK (Fig.7).

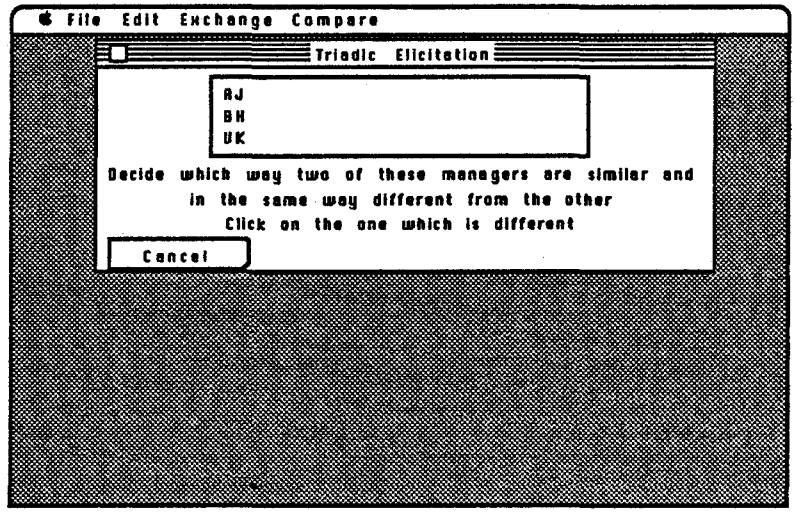

Fig.5 PCS triadic elicitation

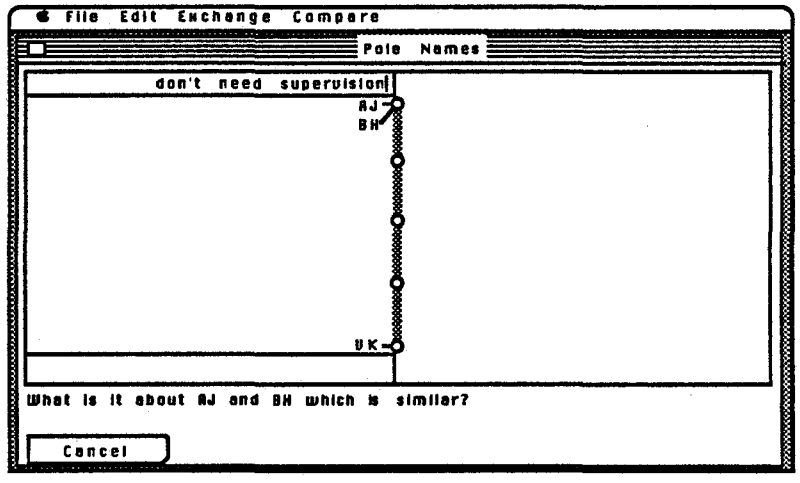

Fig.6 PCS adding pole names

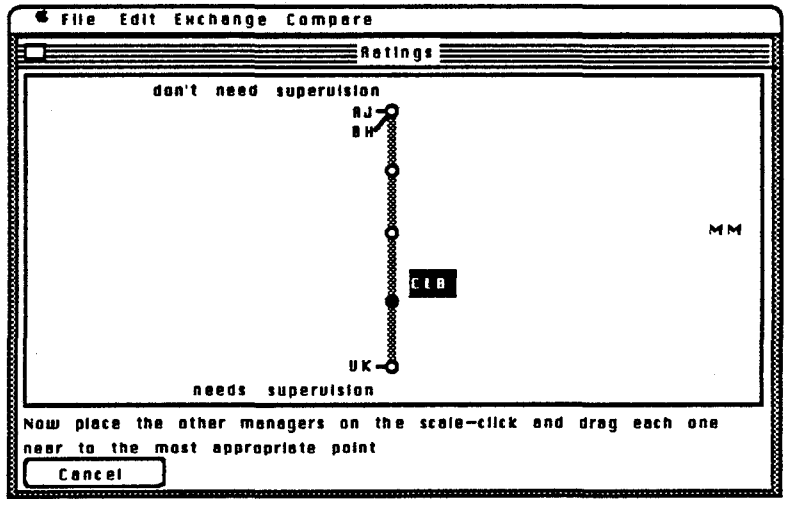

Fig.7 PCS ratings

Now Bill has got his first construct. He now continues adding several more constructs from triads before he decides to look at who else is on the system. He brings up the *Status* window from the *Edit* menu, and sees that Fred has six elements, and Joe has five. He can also see that he has a high construct match, and decides to split it. Perhaps he can use one of Fred or Joe's elements for that (Fig.8).

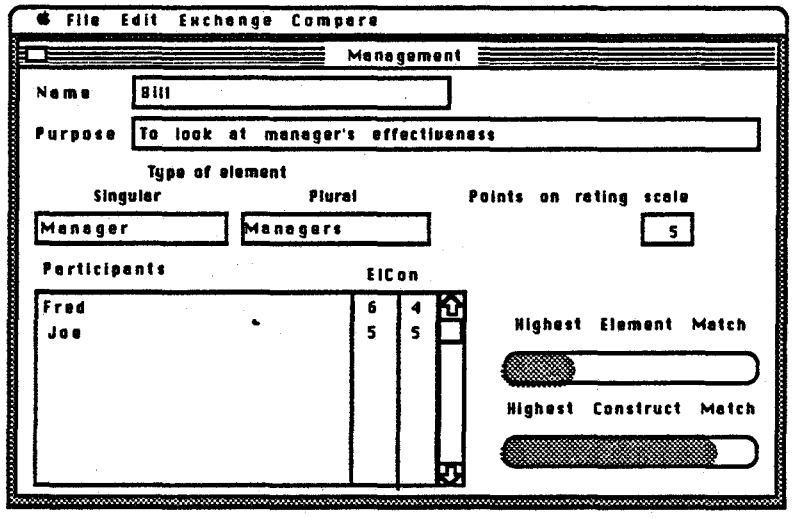

Fig.8 PCS status window

Bill brings up the *Constructs* window from the *Edit* menu and clicks on the *Highest Match* box. This takes him to a new screen showing which constructs are matched (Fig.9).

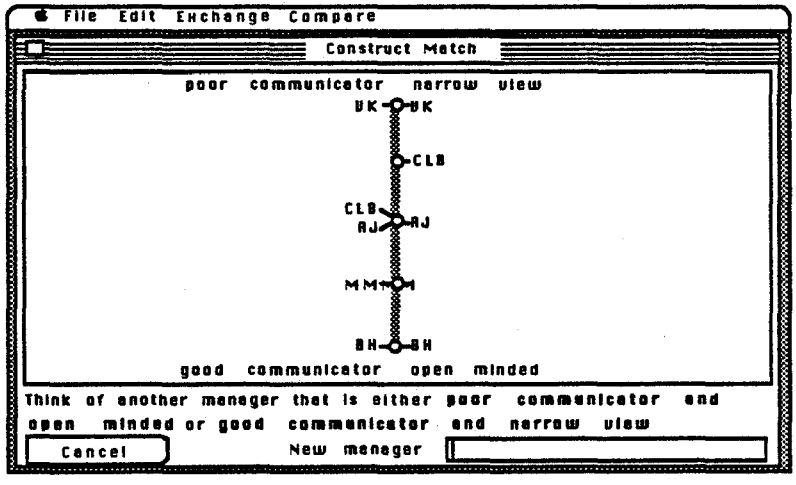

Fig.9 PCS construct match

Bill decides to add an element so he brings up the *Elements* window from the *Edit* menu. Now he can see the elements put in by Fred and Joe, and either click on one of those to add it to his list, or just type in a new one of his own. If he wants to go back and look at the matched constructs again, he would have to go back to the *Constructs* window, and click on the *Highest Match* box. He chooses to add *RA* from Fred and Joe's list (Fig.10).

Next he will have to rate *RA* on each of his constructs in turn. The screen shows his constructs one at a time and he must place *RA* where he thinks he fits on the scale (Fig.11). Note that he may still choose not to rate *RA* on opposite ends of the two matched constructs, but if he does not then the constructs will still be highly matched.

Bill continues to elicit his grid with the options available. At any time he can review his purpose on the *Status* window to make it more closely reflect his choice of elements and constructs. He can add or delete elements or constructs, use triads or just type in names and ratings. When he chooses to finish he can *Close* the file from the *File* menu, and it will be stored as part of the total interaction on the Participant Grid under the topic file name.

There are also two forms of construct *Exchange* available to Bill to enable him to compare his perspective on management effectiveness with those of other participants. He can rate a chosen participant's elements on his constructs either as: Bill would have done; or as Bill thinks the other participant did. These two procedures enable him to compare either his *agreement* with the other participant or his *understanding* of the other participant. He can select either of these forms, or do both one after the other.

Bill chose *Agreement* with Fred and his first screen was that shown in Fig.12. He placed these elements on this construct as if it were one of his own constructs, then continued with the next construct until he had done this for all of Fred's constructs.

To repeat this process for *Understanding* Fred's constructs, Bill had to go back to the *Exchange* menu title, drag to *Fred* as before, then select *Understanding* by clicking on that box. This time he had to rate Fred's elements on Fred's constructs *as he thought Fred had rated them.* The screens looked the same for each construct, just the positioning of the elements was different (Fig.13).

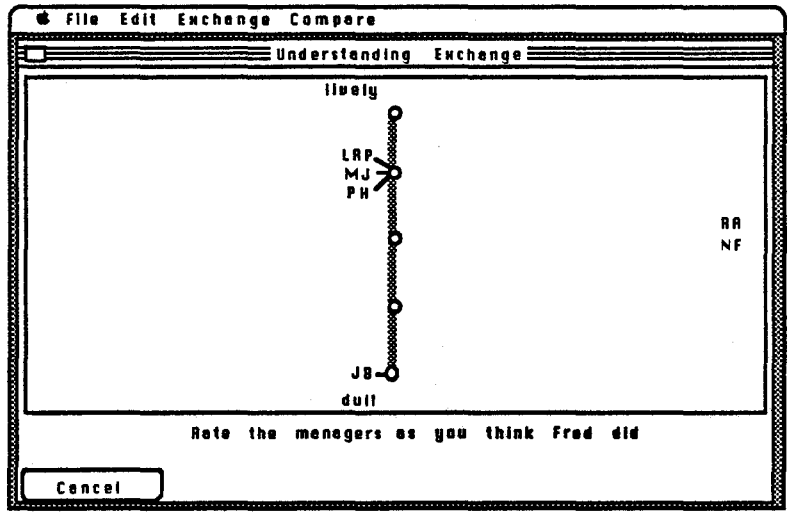

Fig.13 PCS understanding exchange

Conclusions

The new technology of interactive, networked personal computers make it feasible to develop systems for group problem solving, participant systems. This paper has described such a system based on personal construct psychology. The *participant construct system* interactively elicits, exchanges and compares repertory grids from multiple simultaneous users on a network of Macintosh computers. The users of the system can extend and understand their own thinking and problem-solving capabilities by interacting with other people to see the relationship of their points of view to those of others. They can also explore differing terminology for the same constructs; become aware of differing constructs having the same terminology; extend their own construct systems with those of others; provide others with constructs they have found valuable. Thus PCS enables a group of people to explore a problem-solving domain using the full group resources.

Acknowledgements

The development of the participant construct system has been funded by the Alberta Research Council. Financial assistance for the studies on which it is based has been made available by the National Sciences and Engineering Research Council of Canada.

References

Boose, J.H. (1986). Rapid acquisition and combination of knowledge from multiple experts in the same domain. **Future Computing Systems**, In press.

Chang, E. (1986a). Participant systems. **Future Computing Systems**, in press.

Chang, E. (1986b). Participant systems: group human-computer interaction. **Proceedings of 1986 International Conference on Systems, Man and Cybernetics**. To appear. Atlanta (October).

Kasperski, R., Mellen L. & Chang, E. (1986). CANTATA: group protocols in a conferencing environment. **Proceedings of 1986 International Conference on Systems, Man and Cybernetics**. To appear. Atlanta (October).

Kelly, G.A. (1955). **The Psychology of Personal Constructs**. New York: Norton.

Shaw, M.L.G. (1980). **On Becoming a Personal Scientist**. London: Academic Press.

Shaw, M.L.G., Ed. (1981a). **Recent Advances in Personal Construct Technology**. London: Academic Press.

Shaw, M.L.G. (1981b). Conversational heuristics for eliciting shared understanding. Shaw, M.L.G., Ed. **Recent Advances in Personal Construct Technology**. pp. 31-44. London: Academic Press.

Shaw, M.L.G. (1982). PLANET: some experience in creating an integrated system for repertory grid applications on a microcomputer. **International Journal of Man-Machine Studies, 17**(3), 345-360.

Shaw, M.L.G. (1985). Communities of knowledge. Epting, F. & Landfield, A.W., Eds. **Anticipating Personal Construct Psychology**. pp. 25-35. Lincoln: University of Nebraska Press.

Shaw, M.L.G. (1986a). **Participant Construct System User Manual**. ARC-CPCS-PCS-MANUAL:FEB86. Calgary: Alberta Research Council (February).

Shaw, M.L.G. (1986b). Participant systems: group human-computer interaction. **Proceedings of 1986 International Conference on Systems, Man and Cybernetics**. To appear. Atlanta (October).

Shaw, M.L.G. & Gaines, B.R. (1981). The personal scientist in the community of science. Reckmeyer, W.J., Ed. **General Systems Research and Design: Precursors and Futures**. pp. 59-68. Society for General Systems Research (January).

Shaw, M.L.G. & Gaines, B.R. (1983). A computer aid to knowledge engineering. **Proceedings of British Computer Society Conference on Expert Systems**, 263-271 (December). Cambridge.

Shaw, M.L.G. & Gaines, B.R. (1986). Interactive elicitation of knowledge from experts. **Future Computing Systems, 1**(2), to appear.