Human–Computer Interaction — INTERACT'99
Angela Sasse and Chris Johnson (Editors)
Published by IOS Press, © IFIP TC.13, 1999

Speech Interaction Can Support Problem Solving

David Golightly[1], Kate S Hone[2] & Frank E Ritter[1]

[1] School of Psychology; [2] Informatics Institute of Information Technology
University of Nottingham, University Park, Nottingham NG7 2RD, UK.
{David.Golightly,Kate.Hone,Frank.Ritter}@nottingham.ac.uk

Abstract: Speech can lead to increased performance in computer-based problem solving. Studies of complex interaction styles — such as command-line or system-delayed interaction — have found that complex interaction styles can lead to savings in the number of actions users take to reach a solution for certain tasks compared with direct manipulation. Speech can be viewed a complex interaction style because it requires users to reference objects symbolically, rather than through deictic reference, and often involves a system delay. We compared speech interaction with two previously studied interaction styles — delayed interaction and direct manipulation — on the 8-puzzle task. The results show that speech leads to significant savings in moves to solution. Importantly, these savings are not at the expense of greater overall solution times. Savings through speech can be primarily attributed to the effects of system delay. These results suggest the potential for speech interfaces to be used as a tool support problem solving.

Keywords: direct manipulation, system delay, speech recognition, display-based problem solving.

1 Introduction

Speech, the primary and most natural form of human communication, can be used as a mode of interaction. Speech can be used in hands-free situations or as an extra input mechanism, and is used in an increasingly diverse range of applications.

What are the advantages of speech for computer-based problem solving? This paper discusses the cognitive implications of using speech in display-based problem solving. Using speech is not always a neutral interaction style, and can potentially enhance problem solving in certain tasks.

2 Background and Predictions

Display-based problem solving relies on the relationship between represented objects to convey information about the state of the problem (Larkin, 1989). The relationship between interaction-style and display-based problem solving is complex. The common assumption is that the interaction should be made as easy as possible, as interacting using complex interface styles is a drain on cognitive resources. This drain reduces the cognitive resources available to devote to task-related problem solving.

However, anecdotal and experimental evidence suggest that complex interaction is not always disruptive. This introduction reviews the evidence and underlying theory, before presenting the argument that speech, a complex form of interaction, can support problem solving.

2.1 Does Direct Manipulation Suit All Tasks?

Direct manipulation is currently the most pervasive form of interaction. Its success is due to the perceived ease with which users can interact with the display and interpret their actions. Such interfaces reduce the gulfs of execution and evaluation (Norman, 1986) that the user must cross in order to achieve their task goals. Direct manipulation systems accept an input that closely maps to the user's own goal structure, and gives an output that can be easily interpreted in terms of the users own goal structure.

As an example, the user may want to throw away an old text document. The system uses a direct manipulation desktop metaphor. The user can 'pick up' (i.e. point-and-click on) a document and move it into a trash can. The gulf of execution is reduced because the system lets the user act in a way that maps to the user's goal structure. The user acts as if physically manipulating the object. Also, the act of putting it in a 'trashcan' holds a semantic link for the

user with the action of discarding an actual paper file in a waste bin.

The gulf of evaluation is also reduced as the state of the system is immediately interpretable by the user. When the user carries out the action the system gives immediate feedback by moving the file — normally simultaneously with the pointer movement — into the trashcan and removing it visually from the previous file space.

The trashcan example serves to highlight two key aspects of Direct Manipulation. First, direct manipulation uses deictic reference (Ziegler & Fahnrich, 1988). Rather than referencing the objects symbolically by typing a file name, for example, the user points and interacts directly with a representation of the object. The user has a sense of engagement with objects within the system rather than using the interface as an intermediary. The second key aspect of direct manipulation is that it naturally supports reversible actions (Shneiderman, 1992). Errors can be easily undone. In the example above the user can take their file out of the trashcan folder and move it back to its original file space and so the user can try out actions without fear of irreversible consequences.

Overall, the interaction features of direct manipulation are assumed to allow novice or occasional users to understand and explore the functionality of the interface, while allowing expert users to operate quickly. However, there are some situations where direct manipulation may not provide these benefits. Observations from the educational community suggest that the user may be so engaged in the interaction itself, that the learning experience of the interaction becomes marginalized (Moore, 1993). There are many personal accounts of users who feel that command-line languages allow greater flexibility and control of the system or that using command-line enforces a certain degree of 'rigour' on interaction as opposed to almost aimless action with direct manipulation (Clarke, ***INITIALS***, 1998, personal communication). This is not to imply that Direct Manipulation is disruptive *per se*, but it may be the case that, for certain tasks and users, direct manipulation may not be the ideal form of interaction.

A body of work has empirically compared interface styles. Svendsen (1991) and O'Hara & Payne (1998) compared command-line interaction and direct manipulation for problem solving tasks. Svendsen compared the two interaction styles for the Tower of Hanoi task; O'Hara & Payne compared interfaces over the 8-puzzle task (shown in Figure 1). In both cases the command-line users showed significant savings in the number of moves taken to reach the solution state. The command-line participants took longer between moves, indicating greater reflection on the task (i.e. cognitive activity). These longer move times were not so great that it took longer overall to complete the puzzles.

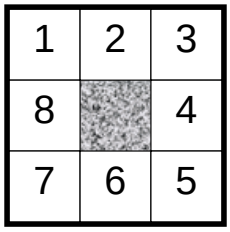

Figure 1: The 8-puzzle. Each tile is moved by sliding the tile into the space (greyed in the figure). The solver starts the puzzle with the tiles randomised and must move the tiles until they match the goal configuration, shown here.

Similar results have been found for an interaction style that was fundamentally Direct Manipulation but with increased complexity. The interaction style used point-and-click interaction, but required the user to click on buttons that represented the tiles rather the tiles themselves (Golightly & Gilmore, 1997). Again, users who solved the 8-puzzle with this style of interaction showed significant problem solving savings over conventional Direct Manipulation users.

These findings can be brought under the heading of Manipulation Supported Problem Solving (MSPS) effects. MSPS is the phenomena of more complex interaction leading to better styles of problem solving. The key elements of MSPS are:

- Problem solving move savings are found with complex interaction styles in comparison with direct manipulation interfaces.
- Longer intervals are taken between moves with a complex interaction style indicating greater cognitive activity.

The following section introduces the mechanism by which complexity leads to shorter solution paths.

2.2 The Cost of an Action Leads to Manipulation Supported Problem Solving

O'Hara & Payne (1998) describe an explanation of why the complex interfaces support problem solving. When the user is faced with a complex interface they carry out a cost-benefit analysis. With the complex interface, executing an action is perceived as having a high cost. Therefore, the user chooses a problem

solving strategy that does not require a high number of costly actions. The strategy the user takes is to plan actions in advance to check that the next move — or sequence of moves — to be executed will move the user towards the goal state.

Consider the case of a command-line interaction style for the 8-puzzle. With a command-line interaction style (or with the interaction style used by Golightly and Gilmore) the user has to refer to each object symbolically rather than by pointing at it. The user perceives a high-cost to having to type in each command many times (or having to click on the button). This encourages the user to plan their actions in advance so that the moves will be effective. Planning results in the longer inter-move times seen with MSPS, but ensures that the user need to take fewer high-cost steps to reach their goal.

On the other hand, Direct Manipulation users perceive a low cost to their actions. As each move is easily executed, the user is prepared to carry out a trail-and-error problem solving strategy. Take the case of a direct manipulation interface for the 8-puzzle. Each move can be made with very little effort, so the user is prepared to explore sequences of moves on the display rather than planning them in advance. The result is that users expend less cognitive effort on each move but take more moves to reach a solution.

There is further support for the 'cost' explanation of MSPS from studies of system delay. The common belief is that system delay has a disruptive effect on interaction. This is often the case with delay leading to reductions in productivity. However, delay has been found to lead to an increase in productivity in some situations — specifically tasks which involve problem solving (see Teal & Rudnicky (1992), for a short review of system delay studies). In such cases users perceive actions that take an extended period to occur in the system as having a high-cost. Rather than having to execute many actions that involve a delay, the user plans moves in advance in order to reduce the number of actions required. O'Hara & Payne (1998) reported that long system delays do indeed lead to fewer moves to solution than short system delays for the 8-puzzle task.

Overall, MSPS occurs when there is some form of interface complexity. The user perceives a cost to their actions. The user tries to reduce the number of actions required by planning actions in advance. This planning leads to longer times between actions but ultimately reduces the number of actions that have to be executed.

The following section introduces the prediction that MSPS should occur with speech-driven interaction.

2.3 Can Speech Interfaces Support Problem Solving?

There have been some studies comparing direct manipulation interaction with speech interaction e.g. (Schmandt et al., 1990) though none have looked directly at problem solving tasks. However, there is good reason for believing that speech could support problem solving. Speech has two types of cost that have lead to MSPS in the past — speech requires the user to symbolically reference objects and speech interaction involves a system delay.

Speech is the ability to symbolically reference objects in the world using sound. It is inherent that speech involves symbolic reference. Work on protocol analysis (Ericsson & Simon, 1993) has found that talking aloud can influence problem solving. Having to refer to objects by name, rather than simply pointing at them, increases the gulf of execution. The user must articulate the name of the object. As Svendsen, O'Hara & Payne and Golightly & Gilmore have shown, symbolically referencing objects when carrying out display-based problem solving invokes a cost. This cost leads to MSPS for the right tasks. Speech should invoke a similar cost, and therefore lead to MSPS for those tasks.

The second source of cost in speech systems is delay. Although increasing processor speed continues to reduce the delay, and continuous speech recognizers reduces it further, the delay is still apparent. Current speech recognition systems still require some time to process speech input. As discussed above, delay disrupts the interaction. In order to minimise this disruption, users plan their actions. Therefore, the cost due to the delay in speech recognition should also lead to MSPS.

Overall, speech requires symbolic reference and objects and incurs a system delay. Combined, these two sources of cost should encourage planning and support problem solving. The following experiment tested this prediction. A direct manipulation interface was compared against a speech interface. The task was the 8-puzzle — a task where MSPS effect had been found in the past. The prediction was that the speech interface should lead to:

- Longer intervals between moves.
- Fewer moves to solution.

A third interaction style, delayed interaction, was also implemented. This was a direct manipulation interface with an equivalent delay to that expected due to the delay due speech recognition. If performance

with the delay was comparable to performance with the speech recognition then this would indicate the problem solving savings through the speech interface were primarily due to system delay. If not, savings are due to the need to symbolically reference objects.

3 Method

3.1 Participants

39 participants (24 women, 18 men) took part in the study. Each participant was paid £5.

3.2 Apparatus

The three interfaces were implemented on a PC with Pentium 166 processor running Visual Basic 5.0. The display had a representation of the 8-puzzle (approximately 10cm $\times$ 10cm) in the centre of the screen. To the right was a smaller example representation of the 8-puzzle already in the goal configuration (shown in Figure 1). On-screen instructions, specific to each interface, told the user how to control the puzzle and that they should get the puzzle to the same state as the example goal configuration.

The Direct Manipulation (Direct) interface required the participant to click on the tile they wished to move. If the tile was next to the space (i.e. the move was legal) the tile would move into the space.

The Speech Recognition interface (Speech) was implemented using Dragon Dictate recognition software. Dragon Dictate is speaker dependent recognition software, primarily for dictation, but was integrated with the Visual Basic project through the Dragon X-tools scripting tool. Instead of clicking on the tile participants said the tile they wished to move by stating the digit on the tile. All participants in the speech recognition condition were trained on the 'intense' setting with the necessary vocabulary (the digits from '1' to '8') before the study. Mean recognition level during the task was 90%.

The Delay (Delay) condition was implemented with a 0.8 seconds delay between the system receiving a mouse click and the tile moving. Testing (by one of the authors) showed that the speech recognition software could be optimised to respond with a minimum system delay of 0.8 seconds. This time included the time required to verbalise the speech input.

There were ten different starting configurations for the 8-puzzle. All participants received the configurations in the same order

3.3 Procedure

Each participant was given a brief introduction to the task. If the participant was in the Speech condition then they trained the recognition software on the task vocabulary.

All the participants were given time to familiarise themselves with the equipment. The participant then read the task instructions, which were clarified by the experimenter, and the participant then started the experimental session in their own time.

The session ended when the participant either completed all 10 puzzles or had worked for 55 minutes, whichever came first. 3 participants in the Delay condition timed-out and were dropped from the analysis.

3.4 Design

The study had a 1×3 between-subjects design varying by interaction style (Direct, Speech, Delay; $n = 12$ for each group). Three measurements were taken from participants:

- Mean move intervals (including the 0.8 second delay for the Delay group). This score was log (ln) transformed to remove the influence of interval peaks in the data. When solution paths are short then a few interval peaks can distort the mean.
- Average extra moves to solution. The minimum number of moves for each puzzle was deducted from the number of moves taken to solve each puzzle. The total for the ten puzzles was then averaged for each participant.
- Total task time.

4 Results

The average move intervals are shown in Table 1. As predicted, both Delay and Speech conditions showed longer move intervals than Direct condition. A 1×3 ANOVA showed an overall effect for interface ($F(2,33) = 30.6$). Post-hoc Scheffé tests confirmed that the Delay and Speech groups had shorter move intervals than the Direct group. Figure 2 shows the move intervals for each puzzle.

The average extra moves are shown in Table 1. As predicted both the Delay and the Speech took fewer extra moves to solution than the Direct condition. A 1×3 ANOVA showed an overall effect for interface ($F(2,33) = 5.7$). Post-hoc Scheffé tests confirmed that the Delay and Speech groups took fewer extra moves to solution than the Direct group. Figure 3 shows the extra moves for each puzzle.

	Delay ($n = 12$)		Direct ($n = 12$)		Speech ($n = 12$)		Significant Differences	Overall
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	$*p < 0.05$; $**p < 0.01$	F (2,33)
Mean move intervals ln(secs)	0.4	0.1	-0.1	0.15	0.27	0.33	** Delay > Direct ** Speech > Direct	** 30.6
Average extra moves (moves)	60.0	25.2	109.8	68.2	53.7	26.2	* Delay < Direct * Speech < Direct	** 5.7
Total time (secs)	1372	416	1522	752	1613	583		0.6

Table 1: Means for each of the conditions across the interactions style (standard deviations given in brackets). Group differences (using a Scheffé test) are given along with overall ANOVA interactions.

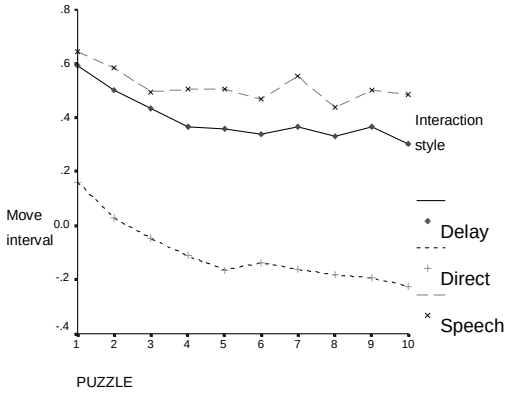

Figure 2: Move latency (ln(secs)) over puzzle

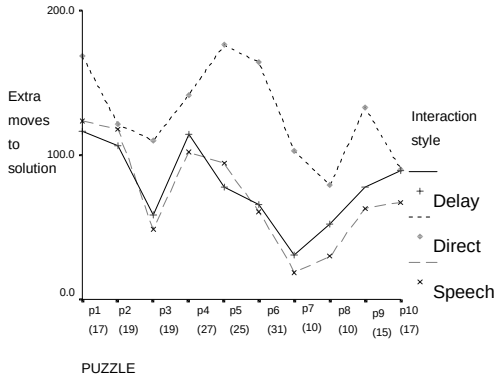

Figure 3: Extra (i.e. extra to the minimum solution path) moves to solution. Numbers in brackets show the minimum solution path for that puzzle.

The total task times are shown in Table 1. There were no differences between conditions. A 1×3 ANOVA showed no overall effect for interface ($F(2,33) = 0.6$). Post-hoc Scheffé tests revealed no group differences.

5 Interpretation and Implications

The results and analysis can be interpreted as follows:

- Delay and Speech interaction users show longer move intervals than Direct Manipulation users. This results indicates that extra cognitive reflection is taking place.
- Delay and Speech users show a saving in the number of moves taken to reach solution.
- There are no reliable differences between the interfaces in the total time to completion for this task.

The results show a significant saving for speech interaction, over direct manipulation, in the number of actions to reach solution for this problem solving task. Though Speech Interaction led to longer move intervals, their cumulative effect was not sufficient to influence total task time. The delay interface user showed a comparable problem solving saving.

Participants are sensitive to the types of puzzles they are solving. Figure 3 shows the extra moves to solution for the series of puzzles given to participants. Changes in the length of the minimum solution path (shown on the abscissa) influences participants behaviour. However, despite this influence, the series of means for Delay and Speech show very similar trajectories. It would appear that the advantage of Speech may be closely related to system delay.

The interpretation that Speech savings are primarily caused by delay is worthy of further investigation. As speech (and CPU) technology improves, the delay inherent in recognition software will decrease. If delay is the primary source of MSPS, then problem solving savings will decrease as recognition speed increases. On the other hand, if the necessity to symbolically reference objects is the primary source of MSPS, then problem solving

savings will remain constant as recognition speed increases. An alternative mechanism for MSPS, Svendsen's 'learning mode' hypothesis (see Svendsen (1991) for an explanation of this mechanism) would predict that savings would remain constant. Clearly, it would be a powerful finding if speech at any speed of recognition supported problem solving.

Whatever the exact cause of MSPS, the effect has occurred here with current technology. The results show that speech interaction can, under the right circumstances, enhance problem solving.

There are several to implications to these findings. First, if one is designing an interface for display-based problem solving, and speech is to be used, then speech may have some influence on how the tasks are performed. For some applications — hands-free situations or by disabled users — there is no option but to use speech. The designer should be aware there is a potential for speech to affect how an application is used.

A more powerful implication is that speech can support the user in carrying out tasks. MSPS effects are most potent for display-based 'transfer' tasks. Transfer tasks are tasks that require many operations to move the problem display through many problem states. If the problem only moves through a few problem states, then there would be no cumulative advantage in extra planning seen in problems such as the 8-puzzle.

Several tasks fall into the category of transfer problems. One category is navigation, either through hypertext systems of through virtual environments. Navigation is carried out by making a series of moves through an information space — particularly in immersive virtual environments. Choices of where to go next are made on the assessing the current state of the information space. The information space is equivalent to the problem space. MSPS involves the user reflecting in order to take the shortest path through the problem space. So speech could be utilised, while a user is navigating, to encourage reflection to take the shortest path through the information space.

Another situation akin to transformation problems is process control. In process control the user may have to constantly alter the state of the system to keep it within certain parameters. MSPS, and therefore speech interaction, would encourage the user to plan before applying an input to the system. The input would be more effective at moving the user towards the goal state, thus requiring fewer inputs.

These are two example applications. Clearly, an important line of research is to verify the advantages of speech and identify the range of applications in which this advantage holds. It is also worth remembering these results indicate that problem solving savings can occur without increasing overall task time. Speech could be of particular value in situations where the number of actions the user can execute is at a premium.

To conclude, speech interaction should not be considered a neutral interaction style. Speech interaction can influence display-based problem solving. Speech can lead to a saving in the number actions taken to reach a goal. This saving is not always at the expense of longer total time spent on the task. There is a potential for speech to support problem solving tasks if the task is right. This potential needs further investigation and exploitation.

Acknowledgements

The authors would like to thank the School of Computer Science and Information Technology, University of Nottingham, for funding this project.

References

Ericsson, K. A. & Simon, H. A. (1993), *Protocol Analysis: Verbal Reports as Data*, third edition, MITP.

Golightly, D. & Gilmore, D. J. (1997), Breaking the Rules of Direct Manipulation, *in* S. Howard, J. Hammond & G. K. Lindgaard (eds.), *Human–Computer Interaction — INTERACT'97: Proceedings of the Sixth IFIP Conference on Human–Computer Interaction*, Chapman & Hall, pp.156–63.

Larkin, J. (1989), Display Based Problem Solving, *in Complex Information Processing: The Impact of Herbert A. Simon*, Lawrence Erlbaum Associates, pp.319–41.

Moore, A. (1993), Siuli's Maths Lesson: Autonomy or Control, *in* J. Benyon & H. Mackay (eds.), *Computers in the Classroom. More Questions than Answers*, The Falmer Press, p.***PAGES***.

Norman, D. A. (1986), Cognitive Engineering, *in* D. A. Norman & S. W. Draper (eds.), *User Centered Systems Design: New Perspectives on Human–Computer Interaction*, Lawrence Erlbaum Associates, pp.31–62.

O'Hara, K. & Payne, S. J. (1998), "Effects of Operator Implementation Cost on Performance of Problem Solving and Learning", *Cognitive Psychology* **35**(***NUMBER***), 34–70.

Schmandt, C., Ackerman, M. S. & Hindus, D. (1990), "Augmenting a Window System with Speech Input", *Computer* **23**(8), 50–6.

Shneiderman, B. (1992), *Designing the User Interface: Strategies for Effective Human–Computer Interaction*, second edition, Addison–Wesley.

Svendsen, G. B. (1991), "The Influence of Interface Style on Problem Solving", *International Journal of Man–Machine Studies* **35**(***NUMBER***), 379–97.

Teal, S. L. & Rudnicky, A. I. (1992), A Performance Model of System Delay and User Selection Strategy, *in* P. Bauersfeld, J. Bennett & G. Lynch (eds.), *Proceedings of CHI'92: Human Factors in Computing Systems*, ACM Press, p.***PAGES***.

Ziegler, J. E. & Fahnrich, K. P. (1988), Direct Manipulation., *in* M. Helander (ed.), *Handbook of Human–Computer Interaction*, North-Holland, p.***PAGES***.

Human–Computer Interaction — INTERACT'99
Angela Sasse and Chris Johnson (Editors)
Published by IOS Press, © IFIP TC.13, 1999

Author Index

Human–Computer Interaction — INTERACT'99
Angela Sasse and Chris Johnson (Editors)
Published by IOS Press, © IFIP TC.13, 1999

Keyword Index