ISSN 1744-1986

Technical Report N° 2004/19

A Simple Taxonomy of Search Reduction in Direct Combination

Simon Holland

10th June 2004

Department of Computing
Faculty of Mathematics and Computing
The Open University
Walton Hall,
Milton Keynes
MK7 6AA
United Kingdom

http://computing.open.ac.uk

A Simple Taxonomy of Search Reduction in Direct Combination

Simon Holland

Department of Computing, The Open University,
Milton Keynes, UK, MK7 6AA
s.holland@open.ac.uk

Abstract. Direct Combination (DC) is a new interaction principle with the capacity to reduce users' search, particularly when interactions involve more than one object or device. The principle applies to arbitrary combinations of physical objects, virtual objects, remote objects, and subparts of objects. In this paper we distinguish between several different ways in which Direct Combination can be used to reduce search, as identified in a preliminary evaluation of a prototype DC interface.

1 Introduction

A characteristic of mobile and pervasive computing environments is that normal use will involve multiple devices working together [1] and multiple objects in the real and virtual worlds [2]. Another characteristic will be dynamic change. People and devices will move between environments several times a day. Environments will have diverse people and devices flowing through them continuously.

Even if we assume that technical configuration were automatic, transparency of use would still remain a distant goal. Dynamicity and interoperability would present significant usability problems, for example in determining what devices, actions and options were available at any moment, and how to specify them. The problems multiply when possibilities are distributed over more than one device: how do you determine what capabilities arise from combining two or more devices, and how do you specify such combined actions?

Mobile devices play important roles in ubiquitous environments, but they typically have resource-poor user interfaces [2, 3] that are difficult to operate for other than simple tasks. Mobile users often have little attention to spare for user interfaces, since there are generally more pressing concerns, such as other people, physical tasks, and physical navigation [2, 3]. It is generally agreed that new user interaction principles need to be devised to simplify the control of complex operations in ubiquitous environments, but there is no agreed solution [4-8].

We propose that a new user interaction principle, called Direct Combination [2, 9, 10], can address a significant part of this problem. The principle is particularly useful whenever two or more objects are involved. Direct Combination applies equally well

to arbitrary combinations of physical objects, virtual objects, remote objects, and subparts of objects. The principle can be implemented using diverse interaction techniques, such as physical pointing [10], speech, and standard selection techniques[2].

The principle affords systematic means for users to reduce their search space when undertaking unfamiliar or complex tasks. In this paper, we will outline the principle, consider examples of its application, and distinguish between several distinct ways in which Direct Combination can be used to reduce search, as recently identified in a preliminary evaluation [2] of a prototype DC interface.

2 The Principle of Direct Combination

The Principle of Direct Combination can be stated in concise form as follows.

- The user interface must always *permit* the user to select *zero, one, two, three or more* objects, before the user is obliged to choose any action.
- The interface must immediately display the actions that apply to *that particular collection* of objects.

Note that the user is *not* obliged to specify one or more nouns before verbs – but this is always *possible*. The Principle of Subsumption, established in [10], requires that DC should always include conventional interaction patterns as special cases.

1 Example scenarios

Let us consider some simple examples of Direct Combination. The following scenarios assume that (unlike today) all devices are wirelessly wide-area networked and so potentially inter-operable. We assume that users can 'select' or refer to remote objects, for example, by selecting an entry on an address book or webpage. For the purposes of this paper, we will ignore security issues and assume that all participants have necessary permissions. The focal problem is to find simple ways to get devices acting in concert to do what is needed.

- 1. Staying in a friend's house, the baby is put to sleep in the spare bedroom. There is no baby alarm, but there is a desktop computer. It would be useful if sound could be piped from the computer to an entertainment centre in the dining room, to act as an improvised baby alarm while the rest of the family eat dinner.
- 2. Driving a hire car equipped with nothing but a car radio, the driver would like to use the time profitably by digesting a briefing document. The relevant file is on a PDA in a briefcase. The only vocalization program the driver can think of to which he has access is built into a speaking teddy in his sons' playroom at home. Is there any way to arrange for the teddy to vocalise the file over the car radio?
- 3. A visitor is working late at night on the top floor in an unfamiliar building, shuttling in between an office and a lab. It was noted earlier in the day that the lock on the main door to the building is broken. The visitor would like to have some kind of warning if the main door is opened so that she can lock herself in the of-

fice. But how to improvise a warning? The visitor decides it would be helpful if the opening of the front door could be set to flicker the room lights.

4 A Simple Taxonomy of Search Space Reduction in DC

The scenarios above are selected from diverse interactions supported by a current implemented prototype DC system. The DC interface controls a simulated ubiquitous environment, with a range of simulated locations and objects, including, cars, door, lights, radios, computers, wall screens, etc with appropriate programmable states and behaviours. The DC server that computes relevant interactions available for different combinations of object types is implemented separately from the interacting objects, though the objects themselves effect the primitive actions The DC system, simulated environment, and server were used in an initial evaluation [2] of Direct Combination to investigate the extent to which DC could offer faster, less frustrating interactions which impose less mental load on the user. This investigation highlighted three distinct ways in which DC can reduce search. We will identify these different kinds of search reduction in turn, and illustrate them by means of the above scenarios.

Choosing one action from actions offered by an object. The first and most straightforward kind of search space reduction is illustrated by scenario 1. In this scenario, the contrast between DC and conventional interaction is relatively simple. Whenever individual objects implement many possible commands, there may be a large space of commands to search. If the user interface lets the user specify one or more objects, or parts of objects, involved in the interaction before having to choose an action (e.g. in this case the computer and the entertainment centre), the system can use the pair of object types as a constraint to offer only the relevant actions. This can be particularly helpful if the devices or user interfaces are complex, unfamiliar, or if the action is infrequently performed. Search space reduction of this kind, among actions in the repertoire of a single device can be dramatic and highly effective [2]. This is the most obvious kind of reduction in search space afforded by Direct Combination, and DC would be valuable if this were the only benefit it offered. In principle, it would be possible, for a given range of tasks, to quantify, using formal models, the reduction in search afforded by DC in this way.

Creating an action composed from several actions from diverse participating objects. This second kind of search space reduction is illustrated by scenario 2. Sometimes, a task that can be achieved using three or more objects in concert *may not be implemented as an integrated command by any one of the objects*. This can happen, for example, when the user interface for some participating devices was designed before some other potential partner device was invented: yet the devices can work in concert to afford new tasks. To carry out such a task, a sequence of actions, perhaps involving different pairs of objects in different steps, may need to be composed by the user. This can be difficult for users to manage using conventional UI architectures. Table 1A shows an example sequence of actions from the recent preliminary empirical study [2]. Without DC, the user must first recognise the need for a sequence of actions, then plan the sequence, and then find the commands needed to

achieve each step (Table 1A). DC interfaces afford a straightforward way for users to access such behaviour in a single step, simply by selecting the relevant objects (Table 1B). (In some situations, a conventional wizard, which typically indexes by *action* or by *task,* may be able to help, but in the situations we are considering, the space of possible *actions* is often too large to search conveniently in this way. Moreover, searching by task generally does not apply to improvised, newly identified, unusually combined, or rarely performed tasks). In any case, whatever approach is used, the most important objects involved typically need to be specified anyway. While DC always allows search by action or task as a special case [10], it offers much richer flexibility in specifying any combination of objects and actions involved.

Table 1. Comparing the steps required to complete task 2, with DC switched on vs. off

User action	***UI Response***
Table 1A: Non-DC Mode	
1 Select *Teddy*	Display commands relevant to *Teddy*
2 Select action *'Vocalize [a vocalizable] using a teddy'.*	Highlight selected command
3 Press *'Do it'*,	*Dialog Box: select a* ***Vocalizable*** *for Teddy to vocalize*
4 Select *text File*	Text file is highlighted
5 Press *'Do it'*,	Dialog Box: *teddy is vocalising the text file.*
6 Select *Teddy*	Display commands relevant to *Teddy*
7 Select action *Pipe sound from Teddy to [a Sound Output]*	Show Dialog Box to choose a **sound output** to pipe sound from Teddy.
8 Select Car Radio	Car Radio is highlighted
9 5 Press *'Do it'*,	Dialog Box *Teddy is outputting sound via car radio*
Table 1B: DC Mode	
1 Select *Teddy*, *text file* and *car radio*	Display relevant commands for object types
2 Select Action *'Vocalize text file using teddy via car radio'.*	Highlight selected command
3 Press *'Do it'*,	Two Dialog boxes show both parts of task completed.

Creating a composed action from diverse participating objects and added functionality from a third party tool. This third kind of search space reduction is illustrated by scenario 3. This interaction requires functionality not associated with any of the participating objects. In fact, in this case, a general purpose *stimulus-response tool* is required to allow the user to choose an action from the door to trigger a chosen response from the light. This is an example where some class of interactions requires additional general purpose functionality. Third party integrative tools and services can provide the diverse kinds of additional functionality needed. However, this can put an additional load on users to know what services and tools are available, what they are called, what they can do, and when they are relevant and required. DC allows users to benefit from such tools and services without needing any knowledge of their existence, simply by selecting the relevant objects. This category differs from categories 1 and 2, because in this case, no amount of composition of two or more operations in the existing repertoire of any of the objects of interest will effect the task. The following comment from a test subject in [2] refers to scenario 2,

but applies broadly to both classes of situation. Asked in a post-evaluation interview to compare the DC vs. non-DC conditions of the evaluation [2], one user responded:

"*If you wanted the teddy to give it a text file in the car radio you could pick all three options that you wanted - you had the freedom to pick them all at once - whereas in the non DC one you really had to use your brain and think of the final output - you didn't have the freedom to pick all three things to put in a box to see what you wanted to do with them*"

Conclusions

In this paper we have made an informal taxonomy of three different kinds of search reduction afforded by Direct Combination, although DC would still be useful if only the first category applied. Other search reduction categories such as *compositions* of category three interactions, documented in [2], and the viewpoint mechanism noted briefly in [10] are beyond the scope of this paper. Other things being equal, reductions in search would be expected to offer interactions which are faster, less frustrating, and impose less mental load on the user. Within the limits of a preliminary study, empirical research supported this conclusion [2].

References

1. Rekimoto, J. (1997) Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments. In ACM Proc UIST 1997. Banff Canada: ACM.
2. Holland, S. (2004) A First Empirical Study of Direct Combination in a Ubiquitous Environment. In HCI 2004, Springer Verlag.
3. Edwards, W.K. and R.E. Grinter (2001) At Home with Ubiquitous Computing: Seven Challenges. In Ubicomp 2001. Springer-Verlag, Berlin.
4. Banavar, G., Beck, J., Gluzberg, E., Munson, J., Sussman, J.B., Zukowski, D. (2000) Challenges: an application model for pervasive computing. Proc. MOBICOM pp 266-274.
5. Weiser, M., (1991) The Computer for the 21st Century. Sc. American. 265-3 p. 66-75.
6. Bellotti,V. Back, M.W. Edwards, K., Grinter, R.E. Henderson, A and Lopes, C. (2002) Ubiquity: Making sense of sensing systems. CHI 2002 pp 415-422.
7. Winograd, T. (2002) Towards a Human-Centered Interaction Architecture. In Carroll, J.M. (Ed.) Human-Computer Interaction.
8. Kristoffersen, S. & F. Ljungberg (2000) Representing Modalities in Mobile Computing: A Model of IT-use in Mobile Settings. Norwegian Computing Center, Oslo.
9. Holland, S. and Oppenheim, D. (1999) Direct Combination. In ACM Proc CHI 1999 pp. 262-269 ACM Press.
10. Holland, S., Morse, D. and Gedenryd, H. (2002) Direct Combination: A New User Interaction Principle for Mobile and Ubiquitous HCI. In Mobile HCI 2002. pp. 108-122 Springer-Verlag.