# *A First Empirical Study of Direct Combination in a Ubiquitous Environment*

**Simon Holland**
Department of Computing, The Open University,
Milton Keynes, UK, MK7 6AA
s.holland@open.ac.uk +44 1908 653148

**In dynamic ubiquitous environments, end users may need to create services by causing *two or more* devices or resources to interoperate together in ad-hoc circumstances. In general, users can find this kind of process hard to manage. At the same time, existing UI architectures are not well suited to supporting such activities. It is proposed that a good basis for addressing these and related problems in a principled, scaleable way is the principle of Direct Combination (DC). The principle is summarized, and analytical arguments are presented that predict that DC can reduce the amount of search required by the user. Other things being equal, such a reduction in search would be expected to offer interactions which are faster, less frustrating, and impose less mental load on the user. We present a proof-of-concept implementation, and a small-scale evaluation of a DC interface. Within the limitations of a preliminary evaluation, consistent support is offered across several measures for the analytical predictions.**

**Keywords:** Ubiquitous Computing, Handheld Devices and Mobile Computing, Input and Interaction Technologies, Interaction Theory, Interaction Design, New interaction principles, New interaction frameworks.

## 1 Introduction and Problem

In Ubiquitous environments, networked devices on the person and in artefacts, vehicles and surroundings, will be cheap, plentiful and richly distributed. In such environments, there will be numerous opportunities for end users to access, or to dynamically create, services of interest by causing *two or more devices or resources to interoperate together,* often under ad-hoc circumstances (Banavar, Beck et al. 2000; Edwards and Grinter 2001; Newman, Sedivy et al. 2002). A very simple, non-problematic example is that a user with a PDA in an unfamiliar place might wish to show another user a document on a nearby screen. In general, users find impromptu interoperation of two or more resources hard to manage (Edwards and Grinter 2001; Kristoffersen and Ljungberg 2000). Existing programming architectures tend make this kind of task inconvenient (Banavar, Beck et al. 2000; Winograd 2002), since the necessary functionality is generally controlled via device-centric application programs (Banavar, Beck et al. 2000). These cannot easily organise the huge number of possible interactions, making it difficult for end

users to cope (Banavar, Beck et al. 2000; Kristoffersen and Ljungberg 2000). Whenever three or more distinct resources are involved, the problems for users multiply combinatorially. Consequently, such tasks often involve the user in non-trivial searches of the user interface. The problems are particularly acute when the search is performed on the move via mobile devices with small, resource-poor user interfaces. Typically users are forced to spend time and attention distracted from their main task, searching a sequence of screens and menus (Holland, Morse et al. 2002). The problem of supporting interoperation in changing circumstances, especially in ubiquitous systems, is called the problem of *spontaneous interaction*. There are few, if any, interaction techniques well suited to spontaneous interaction. Approaches are needed that allow the user to specify what they want as simply and directly as possible, while at the same time taking full advantage of machine-mediated knowledge.

## 2 Proposed Solution

We propose that a good principled basis for addressing this and related problems is the principle of *Direct Combination* (Holland and Oppenheim 1999; Holland, Morse et al. 2002). We will argue that Direct Combination allows the user interface to be made highly economical, and the amount of search required by the user to be reduced. The principle of Direct Combination, with its associated interaction techniques and architecture is perhaps best introduced by means of an example. For reasons of memorability we will use an imaginary interaction scenario borrowed from Holland, Morse et al. (2002) featuring a magic wand. Binsted (2000) has argued that imagined magic is a valuable tool when designing or analysing innovative interaction technologies. More realistic scenarios will be presented below. *Harry raised his wand towards the menacingly advancing Gator*[1] *and tried to remember the spell for turning it into something harmless. It was no good; he just couldn't remember the right spell….* Problems of this sort with magic wands are common in fiction and folklore. For example, the story of the Sorcerer's Apprentice deals with an inexperienced wizard who has memorized enough commands to start a process, but does not know, or cannot recall, the commands needed to control or stop it. *Harry suddenly remembered that this was a Direct Combination Wand. He wouldn't need to recall the spell. Quickly looking around, Harry noticed a small stone on the floor. Pointing the wand at the Gator, Harry selected it, and then made a second* select *gesture at the stone. A glowing list next to the wand presented the two available actions applicable to this particular* pair *of things:* propel the stone at the Gator *and turn* the Gator into stone*. Gratefully Harry activated the second command and the Gator froze into grey immobility*. The key principle of Direct Combination, as illustrated by the above scenario, is that if the user is allowed to indicate in advance *two or more interaction objects* involved in an intended action, then, given a supporting architecture, the system can use this information to constrain significantly the search space of possible commands. This allows the system to present to the user a space of focused relevant options to choose from, instead of the unrestricted space

[1] An imaginary monster, normally docile, but occasionally dangerous.

of commands. Some terminology is worth introducing at this point: interactions involving a pair of objects are known as *pairwise* interactions. This pattern with two interaction objects is particularly useful, but *zero or more* objects are the general DC case. Cases that involve selecting a single object are known as *unary* interactions, and cases with three or more objects are known as *n-fold* interactions. Direct Combination encompasses all of these as special cases within a single uniform framework. Note that any potential difficulties, in some situations, with physical pointing for accurate selection of items, are no barrier to DC; the principle works just as well if objects are identified in any other way, e.g. by speech or by selection from a resource discovery menu.

## 3 The Direct Combination Principle

In practical terms, viewed from the point of view of the user, the principle of Direct Combination (Holland and Oppenheim 1999; Holland, Morse et al. 2002) can be stated 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 under this principle the user is *not* obliged to specify one or more nouns before verbs – it is just that this is always *possible*. The principle of subsumption (Holland and Oppenheim 1999; Holland, Morse et al. 2002) requires that DC should always include conventional interaction patterns as special cases – as illustrated below. To be scaleable and maintainable, appropriate supporting software architecture and analysis methods are required; both of which requirements now have well-founded solutions. These aspects are briefly touched on below, but are outside the scope of this paper. Although DC is at root a very simple idea, it has far reaching consequences.

## 4 The Domain

The domain of the evaluation is a simulated ubiquitous environment, accessible via a user interface (figure 1) running on a laptop controlling a model of the environment. In a fully realized environment, and in earlier work (Holland, Morse et al. 2002), users may select physical objects in the environment by pointing at them physically with a remote ID reader built into a PDA or wand: remote or virtual objects are selected by selecting items on-screen on the PDA/wand. For the evaluation, all selection is done by choosing from menus on a simulated PDA running on the laptop. A wide range of objects are available in the simulated environment, including wallscreens, radios, PDAs, doors, houses, clocks, printers, room lights, cameras, central heating, cars, and text files. Each type of simulated object supports appropriate behaviours and states. Simulated outcomes in the environment are detailed in dialog boxes. Given our interest in interactions that may involve arranging for two or more objects to work together (some possibly remote), the objects in the simulation are drawn from a range of locations including a residence, a place of work, and outdoors (figure 1). Note that DC does not apply merely to physical devices – it applies equally well to arbitrary combinations of physical objects, virtual objects, remote objects, and subparts of objects.

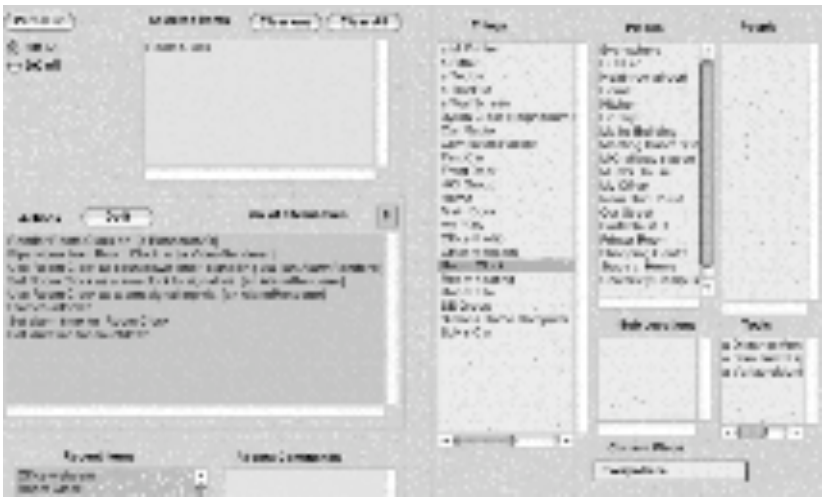

Figure 1. The interface used for the evaluation. For purposes of comparison, Direct Combination (DC) can be switched off. This does not change the appearance of the interface, but restricts its behaviour. DC behaviour always supports not only DC interactions but also conventional interactions (see section 3). The panes *places, people* and *sub-locations* are unused in this study. For reasons of space, the figure has been cropped at the bottom, but this removes no functional detail.

## 5 A Prototype Direct Combination User Interface

The user interface has been implemented to work in two modes, Direct Combination mode, and a reference mode that excludes DC (called for brevity 'non-DC mode'). In the reference mode the interface behaves conventionally, just like most typical user interfaces. Loosely speaking, non-DC mode is as follows: the user may select a single object; this elicits a list of relevant actions; the user may then choose an action from a list, and execute it; this may achieve the task, or it may cause a dialog box to be opened, so that additional arguments can be provided. Of course, many conventional user interfaces have additional behaviors, such as allowing a collection of object to be selected, thus permitting the same command to be issued to all objects in the collection; or cut and paste; and also drag and drop. However, all of these behaviors have analogues in DC (Holland and Oppenheim 1999; Holland, Morse et al. 2002), hence for simplicity and clarity of comparison these behaviors were not included in the evaluation for either condition. In Direct Combination mode, the user interface appears and behaves essentially identically to the reference mode, but has a single systematic *additional* behavior to make it conform to the DC principle: namely, the user may at any time select 0, 1, 2, 3 or more objects of interest. The system then shows a list of actions relevant to that particular combination of object types. As in the non-DC case, the user may select any action and execute it: a dialog box may request additional arguments. This sole behavioral difference is supported by differences in the software architecture 'under the hood', but these differences are invisible to the user.

## 6 Analytical arguments about the predicted benefits of Direct Combination

It is useful to consider some analytical arguments about the expected benefits of Direct Combination. Given a complete enumeration of the options offered to a user by a given user interface(s) in a given environment and circumstances, an abstract tree representation of the space of user commands available can be constructed. In practice, many factors will affect the user's search space, but, given appropriate assumptions about the user, task, and situation, this can be used as the basis of a *formal model* of the user's search space. Such models can be used to estimate the extent to which Direct Combination can reduce the user's search space compared with conventional restricted command patterns, under various assumptions. Here, we will informally outline three general cases in which DC is expected to reduce the search space, and note two other benefits newly identified from this study.

1. Whenever a single object implements a large number of commands, if DC is not available, the user is liable to have a large search space of commands. If the user already knows one or more objects involved in the interaction, DC can be used to shrink the search space simply, rapidly and appropriately.

2. In a conventional interface, whenever choosing from actions involving two or more objects, the user must typically select at least one object in a dialog box. This dialog box both restricts the user's freedom (Holland and Oppenheim 1999), and introduces new visual elements, making new non-task related demands on the user. Such a step is typically eliminated in the DC case.

3. Whenever three or more principal objects are involved in a desired interaction, the combined search space of actions for all three objects considered individually is liable to be large. DC turns the tables to make this combinatorial explosion work in the user's favour. The objects selected reduce the search space to relevant commands only. This is DC's hallmark: a *combinatorial implosion* of the search space, dramatically reducing cognitive load for the user.

4. Sometimes, a task that can be achieved using three or more objects happens not to be implemented as a single integrated command by any one of the objects. Consequently a sequence of actions may need to be composed by the user. This can be difficult for users to manage using conventional application-centric approaches. DC interfaces provide a straightforward way for users to access such behavior in a single step, by simply selecting the relevant objects (see task 3 below).

5. In situations where objects may be capable of interacting in diverse ways with a wide range of other object types, including some that may not have existed when the object was designed, user interfaces would become cluttered and time consuming to search if they dealt explicitly with all possible interactions – and would need continual upgrading to cope with new kinds of interactions. Third party integrative tools are one important way around this problem. The essential idea is that if needed functionality for some class of spontaneous interactions is not available in the user interfaces of any of the relevant devices, a third party tool may be able to afford those interactions.

However, this puts an additional load on non-DC users to know what general-purpose tools are available, what they are called, what they can do, and when they are applicable. By contrast, the role model servers used for DC interactions allow

users to benefit from such tools without needing any knowledge of their existence (tasks 5 and 6), simply by selecting the relevant objects. Automatic use of third party integrative tools can easily be incorporated into DC role model descriptions.

## 7 The Evaluation

This was the first empirical evaluation of Direct Combination applied to ubiquitous computing, and was deliberately kept small scale, since it was unknown what might happen with users who had never used this new interaction technique before.

### *7.1 Assumptions About Subjects*

The evaluation required that all subjects should:

1. Be familiar with the use of *conventional user interfaces*.
2. Be familiar with minimal assumptions about an imagined *simulated world of ubiquitous computing*, as envisaged by Weiser (1991) and others, namely that more or less every object is networked; that a PDA or wand can be used to identify and control such objects, locally and remotely.
3. Have exposure to the fundamental *idea or principle of DC* user interfaces, e.g. as expressed in a few sentences.
4. Have heard one or two *short stories* or scenarios to make the idea memorable.

### *7.2 Subjects*

After two pilot runs with solo users, eight subjects (four pairs of two, to promote think-aloud) were used in the evaluation. The eight subjects were chosen from a pool of about one hundred potential subjects. Within this pool, subjects were recruited with a variety of kinds of computing experience and level of education, and by availability. In order to ensure that all subjects met assumptions 2-4, they were given three sheets to read, as noted in the protocol. All of the subjects except two were male. Three were computer scientists without particular knowledge of HCI. Another was an HCI researcher. Two were secretarial staff with good office computing skills. Two users were non-technically educated schoolchildren (one aged 12 and one aged 17). Both had experience of computer games and word processing, but no particular computing expertise. Two of the computer scientists had heard oral descriptions of Direct Combination at an abstract level. None of the subjects had ever seen or used a Direct Combination interface.

### *7.3 Tasks*

The tasks were drawn from a wide range of actions possible in the simulated environment.

1. Pipe sound from the Office Radio to myPDA.
2. Display time from the Room Clock onto aWallscreen.
3. Pipe sound from myPDA onto the Car Radio.
4. Get Teddy to vocalize a TextFile over the Car Radio.
5. Arrange it so that if the Front Door opens, the Room Light flickers.
6. Arrange it so that when SJH's Car gets closer than 15 miles to Home, the Room Heating is turned on.

For details of how to complete the tasks using DC vs. non-DC, see section 7.7.

*7.4 Data Collection*

Users were observed (with sound recorded, and the screen video-recorded) while carrying out a series of tasks and thinking aloud in pairs. Times taken to complete the tasks were recorded, and whether the task was completed or not. After each task, each subject was asked to grade the task according to mental demand, effort, frustration, and three other dimensions, using the NASA TLX human workload index (Hart and Staveland 1988). After completing the evaluation, users filled in a short questionnaire (see figure 2). The video log was analyzed to catalog all user actions and significant comments.

*7.5 Training*

In order to ensure all subject met assumptions 2-4, subjects were given three sheets to read before the start of each evaluation, each requiring about a minute to read. The first sheet explained that the evaluation takes place in an imagined ubiquitous environment, where every object of interest in the environment is wireless networked and remotely accessible. The second sheet explained that there were two versions of the user interface, and that with one version, only a single object could be selected at a time, whereas with the other version several objects could be selected at once: it was explained that with this version of the interface, when more than one object was involved in a task, selecting more than one object was generally advisable. The third sheet included the Harry Potter scenario and one other scenario, to make the idea memorable. Before starting to perform tasks on either version of the interface, users were read a short scripted screen tour, describing the function of the different panes of the interface, accompanied by the evaluator pointing at the panes in question. No demonstration of the operation of either version was given. The scripted oral screen-tour told users:

- what pane to use to select simulated objects,
- what pane to use to see the objects selected, and
- what pane to use to select relevant commands.

Part of the screen tour varied depending on the version of the interface about to be used (DC vs. Reference), as follows.

*Screen tour for Reference condition.* Users were told that only a single object could be selected from the selection pane at a time. They were also told that in cases where none of the objects individually appeared to be able to carry out the tasks, it might be necessary to use a general-purpose tool. Users were shown the labeled pane where general-purpose tools can be selected. Finally users were told that it may be necessary to use an object they created in an earlier step in order to carry out some tasks, and are told where such objects can be selected (in the pane labeled 'recent items'). Users were told that they could ignore all other panes and controls.

*Screen tour for DC condition.* Users were told that more than one object could be selected at once, and that actions relevant to that particular collection of objects would then be displayed. They were also told how to clear selected objects using the clear buttons. Users were told that they could ignore all other panes and controls. (The panes *places, people* and *sub locations* are unused in this study).

*7.6 Protocol*

Each pair was asked to carry out the same set of six tasks twice: using first one, then the other version of the interface. To allow for learning and interference effects, half of the groups used the DC version first and half used the reference interface first. The protocol followed the methodology of Lewis and Reiman (1993). Users were asked to think out loud to say what they were trying to do next, what they were looking for, what they were uncertain about, etc. After the introductory scripted protocol, the experimenter said as little as possible, except to encourage thinking aloud – any departures from this were recorded. Each task was presented on a new sheet of paper. Tasks were phrased exactly as listed in Section 7.3. After each task was completed, or the subjects gave up, each user was asked to grade each task according to mental demand, effort, frustration, and three dimensions, using the NASA Task Load Index (TLX) (Hart and Staveland 1988). After both conditions had been completed, subjects were asked to fill in a brief questionnaire (figure 2).

| Direct Combination applied to user interfaces tends to: | Disagree Strongly | Disagree | Disagree Weakly | Neutral | Agree Weakly | Agree | Agree Strongly |
|---|---|---|---|---|---|---|---|
| reduce the degree of search required to carry out tasks | | 1 | | 2 | 1 | 4 | |
| reduce the amount of time required to carry out tasks | | 1 | | | 1 | 4 | 2 |
| reduce the amount of attention required to carry out tasks | | | | 2 | 1 | 4 | |
| reduce the amount of work required to carry out tasks | | | | 2 | | 4 | 1 |
| reduce the amount of frustration in carrying out tasks | | | | 1 | 2 | 2 | 2 |
| lessen the need for memorisation in carrying out tasks | | | | 1 | 2 | 4 | 1 |
| lessen the demands of interface navigation | | | 1 | 1 | 1 | 4 | 1 |
| reduce the amount of stress involved in carrying out tasks | | | 1 | 1 | 1 | 4 | 1 |

Figure 2. Subjective questionnaire response, showing number of subjects giving each answer. Modal (most popular) answers are highlighted.

*7.7 Contrasting DC vs. conventional interaction*

The next section contrasts how tasks can be completed using DC vs. non-DC. To make use of the available figures, we start with task 2.

**Task 2.** ***In DC mode,*** the user may select the *Room Clock* and the *Wall Screen* before having to specify any action (figure 3). The system then shows a relatively brief list of actions relevant to that particular combination of objects types. The user selects the action *'Pipe video from RoomClock to WallScreen'*. On pressing *'Do it'*, a dialog box is displayed indicating that video is being piped from the Room Clock to the WallScreen. Note that, following the principle of subsumption, it is also possible to use the DC interface to complete the task using exactly the same steps as with the conventional interface, itemized next.

***In non-DC mode*** the user selects the *RoomClock*, and then selects the action *'Pipe video from RoomClock to [a Video Renderer]'* (see figure 1). On pressing *'Do it'*, the user is presented with a dialog box allowing selection of a Video Renderer - in this case the *WallScreen*. On pressing *'Do it'* in the dialog box, a new dialog box is displayed indicating that video is being piped from the Room Clock to the WallScreen.

**Tasks 1** and **3** are isomorphic in command pattern to task 2.

**Task 4** No single object has enough functionality to be able to complete this task. Two actions must be composed. (Given simple provisos, DC can compose this on the fly, even with objects *types* not explicitly considered before.)

***In DC mode***, the user selects the *teddy*, the *text file* and the *car radio*. The user then selects the option: *'Vocalize a text file using a teddy via a car radio'*

***In Non-DC mode*** the user selects the *teddy*, then chooses the action *'Vocalize [a vocalizable] using a teddy'*. On selecting and executing this action, a dialog box is opened allowing the *text file* to be chosen as argument. A dialog box is then displayed showing that *the teddy is vocalising the text file*. (Recall that no single object supports the UI functionality to complete the task in a single step.)
The user now selects the *teddy* again, and chooses the action *'Pipe sound from Teddy to [a Sound Outputer]'*. A dialog box is opened allowing the *car radio* to be chosen. A dialog box is then displayed showing that *the teddy is outputting sound via the car radio.* This completes the task.

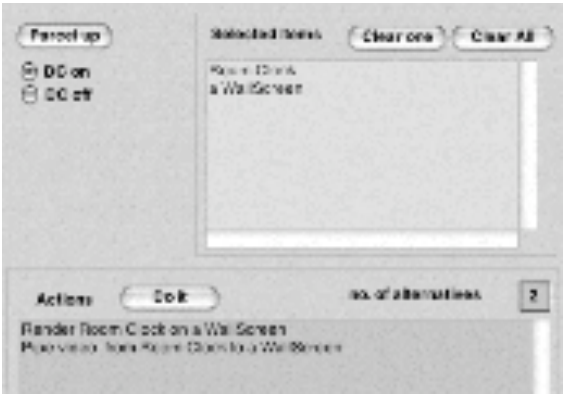


Figure 3. Using Direct Combination. The Room Clock and WallScreen are both selected (task 2). Only actions tailored to that particular set of objects are then displayed. To save space, this figure shows just the relevantly different part of the full interface seen in figure 1.

**Task 5** Again, no single object has enough functionality to allow the task to be performed using a single command. But in this case, there is not even a *composition* of interactions associated with any objects that could complete the task. A general-purpose tool (here the stimulus response tool) must be found and used, as detailed below. (Once more, DC can compose the complete interaction on the fly, even if the particular object types and tools have never been considered together before, and even if the tools are unknown to the participating objects.)

***In DC mode,*** the user selects the *front door* and the *Room light*. From the few choices, the user selects the option: *'Program stimulus response from Front door to room light'*. A dialog box is opened to allow a stimulus to be chosen from those offered by the door (e.g. *open, close, lock, unlock*), and another dialog box to allow a response to be chosen from the room light (e.g. *on, off, flicker*).

***In Non-DC mode***, given that the neither object has the functionality or UI to complete the task, and given that neither object knows about the existence of the third party tool (which may have been designed after both objects were created), the user must select the *stimulus response tool* (see Section 7.5 *Screen tour for DC condition*; and Figure 1, the *Tools* pane). Selecting this tool opens a single dialog box, which allows all necessary arguments to be selected (the stimulus and the response). The user selects the *Front door* as the stimulus source and the *Room*

*Light* as the stimulus responder. A new dialog box is opened to allow the correct stimulus to be chosen for the door, and another dialog box to allow the correct response to be chosen for the room light. This task may seem a little hard, but was deliberately chosen as an example of the class of problem noted under item 5 in section 6. Such situations are common in spontaneous interactions. It is much more easily handled using DC, but it is one of the advantages of DC that it collapses the search space for this kind of problem so decisively.

**Task 6** This task requires not just one, but two general-purpose tools whose functionality must be composed. This is similar to task 6 but more complex. This task was chosen as an example of a class of spontaneous interaction where both situations 4 and 5 described in Section 6 are composed.

***In DC mode,*** the user simply selects the *car*, the *house* and the *central heating*, and then uses dialog boxes as directed to program the details of the condition and response.

**In non-DC mode**, the solution is similar to that of Task 5, but both a Distance Monitor and a Stimulus response tool are required (see section 7.5, screen tour for DC condition, and Figure 1, the *Tools* pane).

## 8 Results

The evaluation was preliminary and small scale. However, the combination of measures used (four NASA TLX measures, task completion times, whether tasks could be completed or not, and the questionnaire) allowed triangulation of results. Figure 4 shows the *Mental Load* of the subjects as measured on a 20-point scale by the NASA TLX (Hart and Staveland 1988), averaged over all eight subjects, in the DC condition compared with the non-DC condition, for the six tasks. As figure 4 shows, the *Mental Load* for the non-DC condition was measured to be bigger for all of the six tasks. Figures 5 and 6 show the same information for *Effort* and *Frustration* as measured by the NASA TLX. Both *Effort* and *Frustration* were measured to be greater in the reference condition than in the DC condition for all of the six tasks. This is also true of the NASA TLX results for Physical Load and Temporal Load (not shown). Figure 7 shows the actual time taken to complete each task (or until the participants abandoned the task). The time taken, averaged over all subjects, was longer for the non-DC condition than the DC condition for each of the six tasks. Given the think aloud nature of the evaluation, time taken must be treated with caution, but, within the limits of a preliminary study, it gives a useful indication. All of the subjects completed all tasks in the DC condition, but in the non-DC condition, two subjects gave up on one task, and two subjects gave up on two tasks. In the questionnaire (Figure 2), seven of the eight subjects rated the DC condition favorably, or at worse, neutrally on all of the questions asked.

### *8.1 Effect of Order of Presentation*

As already noted, half of the subjects encountered the DC condition first, and half encountered the non-DC condition first. Irrespective of the order of presentation, averaged over all tasks, all subjects rated the DC tasks on the NASA TLX scale as having a lighter workload than the non-DC tasks for: mental load, effort, frustration, physical load and temporal load. Similarly, averaged over all tasks, all subjects performed the tasks faster on the DC than on the reference interface.

However, the order of presentation did affect the *degree* of out-performance of DC over non-DC differently for different measures and different tasks.

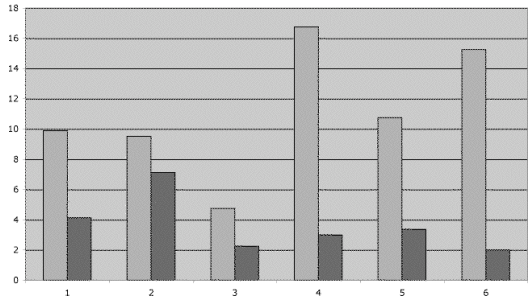


Figure 4. Mental Load for DC vs. non-DC, on a 20-point scale, as measured by the NASA TLX (averaged over all subjects) for tasks 1-6. DC scores are shown in the darker shade.

# 9 Interpretation

*9.1 Principal Findings*

This is the first empirical evaluation of a DC interface applied to ubiquitous computing, so we were unsure what user behavior might be found. Consequently the use of a small-scale evaluation was chosen. One problem is that eight subjects are not enough for reliable statistical evidence. Also, think-aloud activity, though useful here (see 10.3) confounds precise timing data. Consequently, we must forgo any *precise* claims about timing. However, the timing data *overwhelmingly* favoured DC (figure 7), even though no user had ever used it before – which might be expected to handicap DC. Given that all of the several data sources yielded similar stories, we claim that, within the limits of a preliminary evaluation, the study gives useful information. On the basis of the results from the various NASA TLX dimensions, the timing data, the completion data, the order of presentation analysis, and the questionnaire, there is consistent evidence, within the limitations of an initial study, that Direct Combination has the capacity to offer interactions which are faster, less effort, less frustrating, and impose less mental load on the user. There was support for all of the analytical predictions (section 6).

*9.2 Order of Presentation*

The above findings held true irrespective of order of presentation, although the *degree* of difference of the various measures varied with the order of presentation. However, the degree and direction of variation was inconsistent across the various different measures, so that it is hard to draw firm conclusions about this variation.

*9.3 Direct Combination and Task Complexity.*

Unlike in non-DC, on average the more complex task 4 was completed in the DC condition in similar times to the simpler tasks 1-3. Even tasks 5 and 6 did not take much longer than task 1 in the DC case, even though both conditions involved the same dialog boxes for setting up trigger distances and stimulus responses. DC comfortably outperforms non-DC even for the three simple tasks 1-3, although performance with non-DC does appear to improve over the course of these tasks.

*9.4 Learning to Use Direct Combination*

Given that all subjects had hundreds of hours of experience with conventional user interfaces, which work more or less exactly like the reference interface, and no experience at all of a DC interface, it was not clear in advance that users would find DC easy to use or quick to learn, especially since users had only about two or three minutes of introduction to DC. However the results demonstrate, as did direct observation, that learning to use a DC interface was almost instant.

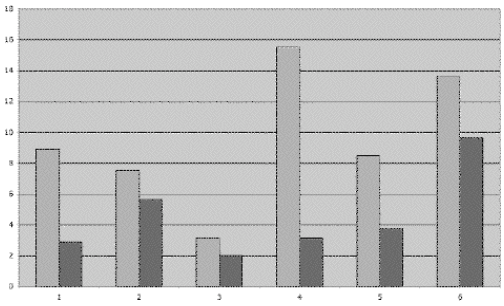

Figure 5. Effort for DC vs. Non-DC, on a 20-point scale, as measured by the NASA TLX. Average ratings over all subjects for tasks 1-6. DC scores are shown in darker shade.

## 10 Alternative interpretations & limitations

*10.1 Minor Inconsistencies: Questionnaire Results*

In the questionnaire, one subject (the HCI researcher) had evaluated DC negatively on several measures (figure 2), even though, in the NASA TLX, he consistently rated the DC condition better for each task. The key to this apparent self-contradiction was identified when we asked this subject what his responses would have been to the questionnaire with *negated* versions the questions, i.e. asking about the effect of "the *exclusion* of Direct Combination from user interfaces". Surprisingly, the *negated* versions of the questions elicited exactly the same responses, *except* for the question about memorization. The subject agreed that the tasks had been easier, quicker, and less stressful, etc using DC: his negative responses were not about DC *per se*, but reflected his skepticism about any general statements about interaction techniques. Despite systematic skepticism about general claims for *any* interaction style, this subject actively supported the view that Direct Combination tends to lessen the need for memorization (figure 2).

*10.2 Minor Anomalies: Workload in First Tasks*

On all of the workload measures, task 2 in the DC condition shows a higher load than tasks 1 and 3, despite the fact that tasks 1-3 are identical in command structure. There was an accidental ambiguity in the written phrasing of task 2, which caused all subjects *in both conditions* to be unclear which of two similar actions best fitted this task. The additional time taken in task 2 compared with task 1 is accounted for in the video log by time exploring which of the two actions better satisfied the task. This was observed in both conditions, but in the non-DC condition, the effects seem masked by other factors, possibly learning effects.

*10.3 Usability Issues*

An earlier heuristic review had removed as many usability problems as possible in both conditions. Because the DC interactions are relatively simple and undemanding on the user interface, most of this earlier improvement effort had focused on the non-DC version of the interface. For example: care was taken with non-DC to avoid a sequence of dialog boxes where a single combined dialog box could be used; a 'show matches only' feature was added; and the number of general purpose tools was reduced to only those needed for the tasks, plus a single distractor. Despite this, the think-aloud element of the present study picked up numerous small usability features that could be improved for both conditions equally: for example, the workflow could have been better organized left to right, and some of the 'success' dialog boxes were insufficiently detailed. Three of these issues may have affected the non-DC side more than the DC side: actions were not categorized; the 'show matches only' feature for selecting arguments in dialog boxes was not set as the default mode; and the general purpose tools could have been more clearly named. However, study of the video log together with a simple analytical argument (below) suggests that these issues made little material difference. Firstly, the video log suggests that setting 'show matches only' as default could have had only a small impact on overall timings. Similarly, although some of the users expressed confusion about the names of the general-purpose tools, there was no such confusion on the part of other users, and yet there was no material difference in the outcomes. Finally, the first argument in the analytical predictions section shows that simple categorization of actions is soon outrun by a combinatorial explosion of possible object interactions – so that action categorization can help non-DC interactions only to a limited degree.

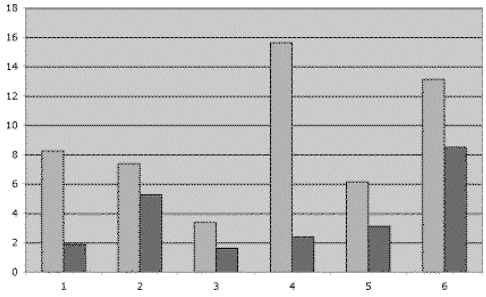


Figure 6. Frustration for DC vs. non-DC as measured on a 20-point scale by NASA TLX (average ratings for all subjects) for tasks 1-6. DC scores are shown in the darker shade.

## 11 Other issues

In previous studies, the question was raised whether DC would transfer workload from the user to the domain analyst and maintainer (Holland and Oppenheim 1999; Holland, Morse et al. 2002). There were also at that time technical difficulties with n-fold combination. Although outside the scope this paper, it is worth noting that two innovations, the use of a role-based architecture, and the use of computationally explicit role models, as used in the current implementation give well-founded solutions for both concerns. The architecture appears scaleable, flexible and potentially well suited to distributed use. Various issues have been

identified that must be addressed for the general applicability of *any* user interaction techniques in ubiquitous domains (not just DC) (Edwards and Grinter 2001; Bellotti, Back et al. 2002). These issues include security, resource discovery, feedback, monitoring of tasks in progress, and canceling. We have implemented crude prototype facilities for monitoring of tasks in progress and canceling, but for simplicity these facilities were hidden for the purposes of evaluation. Note that this is a completely new architecture compared with Holland and Oppenheim (1999) and (with sound n-fold combination) a leap beyond Holland, Morse et al. (2002). As regards scalability, DC places no burdens on distributed objects beyond those typical for Ubiquitous Computing. i.e. simply to identify their class or identity, and to respond to commands. Similarly, only a very simple client is required on the user devices; all of the heavy lifting can be done by DC servers. Also, because of the use of reflective descriptions, server requests require only modest bandwidth (not complete descriptions of objects).

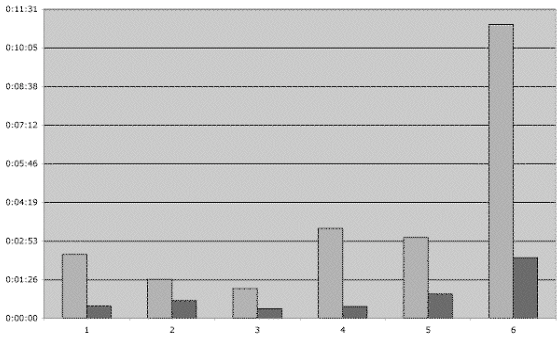


Figure 7. Time in minutes and seconds to complete tasks for DC vs. non-DC. Averages for all subjects shown for tasks 1-6. (Max time about 11 minutes). DC in darker shade.

## 12 Related work

The Direct Combination principle may be viewed as generalizing and extending diverse existing user interaction approaches in a parsimonious, elegant way. Use of pointing devices to transfer information between computers or other devices is well established. For example, Pick-and-Drop (Kohtake, Rekimoto et al. 1999; Rekimoto 1997) is an extension of Drag & Drop used to copy data between multiple devices via passive pens with IDs. Direct Combination may be viewed in turn as an extension of Pick and Drop, but one that offers far greater flexibility and expressiveness. Similarly, the InfoStick (Kohtake, Rekimoto et al. 1999) is an interaction device for inter-appliance computing. It may be used to pick up and store information items between devices. The InfoStick effectively offers a limited special case of Direct Combination where the only available operations are *get* and *put.* Previous work on ad-hoc configuration includes the Proem project, which aimed to provide infrastructure for building special-purpose ad-hoc collaboration applications (Kortuem 2002). The Aura software architecture (Sousa and Garlan 2002) also addresses dynamically changing resources, but it centers on technical challenges, whereas our concern is on the level of user interaction. The tangible computing system DataTiles (Rekimoto, Ullmer et al. 2001) uses tagged

transparent tiles placed on a flat display. Interactions between any two tiles are effected by physical adjacency, or by a pen gesture. The kind of interaction is determined by the tile types, although a pen gesture may be used for limited modifications. However, the affordances are very limited - physical adjacency determines a single interaction type. DataTiles could be given greater flexibility and power, without loss of elegance, by applying the Direct Combination principle. Alternatively, DC may be viewed as a novel way of exploiting a *relational* approach (Ullmer and Ishii 2000) to Tangible User Interface design, *systematically* allowing the selection of multiple objects to determine dynamically bindings between objects and computational operations. Other systems with related goals, but different approaches include Recombinant Computing (Edwards, Newman et al. 2002), and the iRoom tuple-space approach (Borchers, Ringel et al. 2002). DC relates strongly to Direct Manipulation: parts of DC may be viewed as generalizations or specializations of DM, though the relationship is complex.

## 13 Conclusion

Direct Combination is a new user interaction principle. Fragmentary, isolated examples of DC can be identified in some existing systems, but as a *systematic principle* and supporting framework, Direct Combination is fundamental and novel. This paper has presented the first systematic, albeit preliminary, empirical investigation of a Direct Combination user interface in the ubiquitous domain. The investigation is small scale and the results must be treated cautiously, but across all data sources there was consistent preliminary evidence that a DC user interface, compared with a conventional user interface offers interactions which are faster, less effort, less frustrating, and impose less mental load on the user. Within its limits, the study also demonstrated that DC is usable for users from a variety of backgrounds, and is rapid for them to learn. Building the test environment led to the identification of three new benefits of DC for users, beyond reduction of search space, not previously explicitly noted. These concern automatic composition of actions, the automatic deployment of integrative tools not associated with specific objects, and the automatic *composition* of the operation of such tools (see items 4 and 5 in section 6). One surprising lesson arising from this study was that, once the architecture was in place there was considerably more work needed to implement the programming required for the *conventional* interactions than for the DC interactions. This was because, to eliminate any factors that might unfairly disadvantage the conventional interactions, tuning was carried out *solely in the non-DC case* to ensure that all dialog boxes would be as clear as possible. For the DC case, no such dialog boxes, or tuning, were needed, so that work disappeared not only for the user but also for the designer/developer. No comparable tweaking was done for any of the DC interactions – they were all done using the standard elements of the role-based DC architecture. Within the limits of a preliminary evaluation, this study tends to substantiate the theoretical arguments for the benefits of DC. No evidence emerged of undue penalties that have to be paid elsewhere. More generally, the evaluation suggests that the principle is very widely applicable to user interfaces. The principle is particularly useful when it is necessary for an end-user to arrange for two or more devices or resources to interoperate together in ad-hoc circumstances.

## Acknowledgements

Thanks to Paul Mulholland for generous and vital advice, Bashar Nuseibeh for urgency, Marian Petre for insightful tips, SJH for vital help, Henrik Gedenryd for the UC connection, David Morse for support, and to the anonymous referees for much appreciated comments and criticisms.

## References

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.

Bellotti,V. Back, M.W. Edwards, K., Grinter, R.E. Henderson, A. Lopes, C. (2002) Ubiquity: Making sense of sensing systems. CHI 2002 pp 415-422.

Binsted, K. (2000) Sufficiently Advanced Technology: Using Magic to control the world. CHI 2000, pp 205-206.

Borchers, J., Ringel, M. Tyler, J. Fox, A. (2002) Stanford Interactive Workspaces: A Framework for Physical and Graphical User Interface Prototyping. *IEEE Wireless Communications*.

Edwards, K. and Grinter. R. (2001) At Home with Ubiquitous Computing: Seven Challenges, Ubiquitous Computing 2001.

Edwards, K., Newman, M.W., Sedivy, J., Smith,T., Izadi, S.(2002) Challenge: Recombinant Computing and the Speakeasy Approach. *Proc Mobicom '02*.

Hart, S., Staveland, L. (1988) Development of NASA-TLX (*Task Load Index): Results of empirical and theoretical research*. In Human Mental Workload, Hancock, P., Meshkati, N. (Ed.), 1988, pp 139 - 183.

Holland, S. and Oppenheim, D. (1999) Direct Combination. In Proceedings of ACM CHI Addison Wesley ISBN 0-201-48559-1. 1999, pp. 262-269.

Holland, S., Morse, D.R., Gedenryd, H. (2002) Direct Combination: a New User Interaction Principle for Mobile and Ubiquitous HCI. In Paterno, F., (Ed.) (2002) Human Computer Interaction with Mobile Devices. Mobile HCI 2002, Springer Verlag LNCS pp. 108-122.

Kohtake, N., Rekimoto, J. and Anzai, Y. (1999) InfoStick: an interaction device for inter-appliance computing, HUC 99.

Kortuem, G. (2002) Proem: A Middleware Platform for Mobile Peer-to-Peer Computing. ACM SIGMOBILE Mobile Computing & Communications Review (MC2R), Vol. 6, 4.

Kristoffersen, S. & Ljungberg, F., (2000) Representing Modalities in Mobile Computing: A Model of IT-use in Mobile Settings. White Papers, Norwegian Computing Center, Oslo.

Lewis, C. and Reiman, J. (1993) *TaskCentered User Interface Design: A Practical Introduction*. University of Colorado, Boulder, CO, 1993.

Newman, M.W., J.Z. Sedivy, W.K. Edwards, T. Smith, K. Marcelo, C.M. Neuwirth, J.I. Hong, and S. Izadi. (2002) Designing for Serendipity: Supporting End-User Configuration of Ubiquitous Computing Environments. *(DIS2002)*.

Rekimoto, J., (1997) Pick and Drop: A Direct Manipulation technique for multiple computer environments, Proc UIST 97 pp. 31-39.

Rekimoto, J., Ulmer B. and Oba, H. (2001) DataTiles: A modular platform for mixed physical and graphical interactions. In Proc CHI 2001. 2001 p 269-276.

Sousa, J.P., Garlan,D. (2002) Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments. WICSA 2002: pp. 29-43.

Ullmer, B., Ishii, H., (2000) Emerging Frameworks for Tangible User Interfaces. IBM Systems Journal 39 (3&4), pp. 915-931.

Weiser, M. (1991). The Computer For the 21st-Century. Scientific American 265(3): 66-75.

Winograd, T. (2002) Towards a Human-Centered Interaction Architecture. In Carroll, J.M. (Ed.) (2002) Human-Computer Interaction.