The application of Direct Combination to Mobile and Ubiquitous Human Computer Interaction.

Simon Holland, David R. Morse, Henrik Gedenryd

Computing Department, The Open University, Walton Hall,
Milton Keynes, MK7 6AA, United Kingdom
Fax +44 1908 652140
TR 2002/1
{D.R.Morse, H.Gedenryd, S.Holland}@open.ac.uk

A shorter, adapted version of this paper was submitted to Mobile HCI 2002.

The application of Direct Combination to Mobile and Ubiquitous Human Computer Interaction.

Simon Holland, David R. Morse, Henrik Gedenryd

Computing Department, The Open University, Walton Hall,
Milton Keynes, MK7 6AA, United Kingdom
Fax +44 1908 652140
TR 2002/1
{D.R.Morse, H.Gedenryd, S.Holland}@open.ac.uk

Abstract. Direct Combination is a recently introduced user interaction principle. The principle (previously applied in the desktop computing context) and its associated techniques have the capacity in certain circumstances to reduce significantly the degree of search required to operate user interfaces. In this paper we argue that the Direct Combination principle (DC) applies particularly aptly to mobile computing devices, given appropriate interaction techniques, examples of which are presented here. We argue that the reduction in search afforded to users can be applied to address several outstanding issues in mobile and ubiquitous user interaction including: limited feedback bandwidth; minimal attention situations; and the need for ad-hoc spontaneous interoperation and dynamic reconfiguration of multiple devices. The application of Direct Combination to mobile and ubiquitous interaction allows the user to exploit objects in the environment to rapidly narrow down the range of interactions that need be considered (by system and user). Ambient Combination may thus be viewed as a new way of distributing the user interface in the environment. When the DC technique of pairwise combination is applicable, it can greatly lessen the demands on users for memorisation and interface navigation, and can facilitate the use of recognition as opposed to recall when it is necessary to specify unfamiliar commands. Direct Combination, when extended and adapted to fit the demands of mobile and ubiquitous HCI is referred to as *Ambient Combination (AC)*. In this paper we present Ambient Combination through a series of interaction scenarios. We describe a prototype Ambient Combination system, and the results of using this prototype system in a number of scenarios. Brief notes are given on the architecture. We discuss the principle of *subsumption* that allows Direct Combination to stay out of the way of users when it is not useful. We briefly discuss how preliminary analysis of the new abstract achitecture suggests that Ambient Combination offers a new approach and new leverage in dealing with context-aware interactions. Almost all of the *elements* of Ambient Combination

exist in fragmented or partial form in other systems: what is new is the simple uniform framework and approach that DC affords for dealing with challenging interactions, and the abstract architectures that make this possible.

1 Introduction: problems with mobile HCI

User interfaces for mobile devices must typically deal with four general problems. Firstly, only limited screen real estate is available; more generally, taking into account non-visual forms of feedback such as auditory displays, *feedback bandwidth* is limited. Secondly, the bandwidth, precision and convenience of *input devices* are generally restricted. A third problem is that many mobile devices are typically used in *minimal attention situations* (Pascoe et al., 2000), where the user has only limited, intermittent attention available for the interface: in such situations, interactions with the real world are generally more important than interactions with the computer; the users hands and eyes may be busy elsewhere; and the user may be busy avoiding the normal hazards of moving around, as well as engaging with real-world tasks. Fourthly, as devices diversify and proliferate, users increasingly face the need to make two or more devices *interoperate for some ad-hoc purpose*. Even where each device has a well-designed user interface, this kind of task can be hard to arrange.

These four factors, singly and in combination, can cause difficulties in situations where the user does not know, cannot recall or cannot locate the commands needed to make the computer carry out a desired action. For mobile devices and their typical contexts of use, it is often inconvenient, impractical or too time-consuming to navigate to the appropriate screen or to otherwise search the space of available commands to effect the appropriate action.

In the case of context-aware devices, a final cluster of problems applies: there is no generally agreed uniform framework for applying contextual information. Context-aware systems tend to make incorrect guesses; and the user generally has little scope for correcting or capitalizing on incorrect guesses (Erickson, 2002). This paper argues that Ambient Combination has a part to play in addressing all of these problems, and has a useful role in any future framework for mobile and ubiquitous HCI.

2 Ambient Combination: Direct Combination applied to mobile HCI

The four general problems related to mobile user interaction noted above can be viewed as problems of *search*. That is to say, they all cause problems whenever a mobile device does not immediately afford the currently desired action, and users are consequently forced to navigate the interface, drill down, scroll or otherwise search the interface for the item needed to perform the intended action.

Direct Combination (DC) [Holland and Oppenheim, 1999], previously applied to desktop computing can often significantly reduce the degree of search required to

operate user interfaces. In this paper, we argue that Direct Combination can be applied even more aptly to mobile computing and inter-device interactions. We will use the term *Ambient Combination* to indicate adaption, extension and application of DC principles to Mobile and Ubiquitous Computing.

One convenient way to introduce Ambient Combination is by means of an imaginary interaction scenario featuring a magic wand. Binsted (2000) has argued that imagined magic can be a valuable source of inspiration when designing innovative forms of technology. We will use a hypothetical magic scenario to illustrate a key usability problem with magic wands (and other mobile devices). We will then illustrate how Direct Combination can address it. Later in the paper we will present non-magic interaction scenarios that we have implemented in prototype.

2.1 A hypothetical scenario: Harry and the DC Wand

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 memorised enough commands to start a process, but does not know or cannot recall the commands needed to stop it.

Harry suddenly remembered that this was one of the new Direct Combination Wands. 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 made the select *gesture 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.*

One insight of Direct Combination is that if the user is allowed to indicate in advance two or more interaction objects involved in an intended command, the system can often use this information to constrain significantly the search space of possible interactions. This allows the system to present to the user a space of focused relevant options to choose from, instead of the unrestricted space of commands. In an environment rich in objects of interest, users often know what objects they want to use (a printer, a wallet, a car, a door, a document etc) - but commands, apart from commonly used ones, tend to be relatively more abstract, and harder for people to recall. The contrast is between *recognition* (easy) and *recall* (hard). Given this insight, Direct Combination user interfaces give the user the freedom to specify the

[1] An imaginary creature.

parts of commands in any order desired, for example *noun noun* (to be followed later by a verb), or the more conventional *noun verb*. At each stage, feedback is given on how this constrains the available options (examples follow later).

However, while this kind of pairwise interaction is sometimes useful, it is not useful all the time. For example, sometimes the objects that would help to constrain the space of relevant actions are not to hand in the environment. Hence Direct Combination requires that pairwise interaction always be made available in such as way that in does not get in the way conventional interaction methods or established practices. Direct Combination offers more ways of getting things done, but should never leave the user worse off than if DC was not provided - conventional methods should always be available. This can be illustrated by replaying Harry's imaginary encounter in more detail.

Harry raised his wand towards the menacingly advancing Gator and tried to remember a spell for making it harmless. Harry could see in glowing letters next to the wand the most common general spells, and category titles grouping all known spells, but he had no time to search them. Pointing the wand at the Gator with the select *gesture, the list of general spells vanished, to be replaced by a shorter list of the most common spells applicable only to Gators. The entire list of Gator-specific spells was now available - but Harry had no time to deal even with this list. Remembering at last that this was a DC compliant wand, Harry* made a *second* select *gesture at the stone. The glowing list next to the wand shrunk to the two available actions applicable to this particular pair of things:* propel the stone at the Gator *or* turn the Gator into stone. *Gratefully Harry activated the second command.*

The list of Gator-specific commands that appears when Harry selects the Gator alone is nothing new to DC enabled devices: many existing systems have such a feature. Even less remarkable is the fact that the DC wand would have allowed Harry to point at the Gator and then invoke the verb (spell) directly, if his memory had permitted. Although Direct Combination allows users to employ pairwise interaction *(noun noun…)* whenever they wish, the user is never forced to work in this way. A DC command processor should accept pairwise interaction along with more conventional ways of specifying commands such as *noun verb* and *verb arguments*. Indeed the Direct Combination principle of *subsumption* (explained later) requires this freedom. DC also requires the system to provide appropriate feedback at all points that the user partially specifies the command, in order to show how choices so far constrain or suggest further choices.

The power of Direct Combination is greatly enhanced when the assignment of operations to pairs of objects need not be specified exhaustively by the designer but can be propagated by inheritance from definitions in carefully factored abstract

classes in appropriate class hierarchies, either held locally or in a more distributed fashion (Holland and Oppenheim 1999).

3 Scenarios illustrating Ambient Combination

We have informally illustrated two principles of Ambient Combination, pairwise interaction and subsumption. Ambient Combination is most easily understood further by analysing more detailed scenarios.

3.1 Technology assumed for scenarios

Figure 1a shows the simulated screen of a small PDA, such as a Palm Pilot, iPAQ or Psion. The PDA has a stylus for carrying out detailed work, but can also be operated one-handed for many simple purposes by means of the buttons provided. Plugged into the system is a scanner for reading ID-tags indicating the *object* identity or *class* identity of devices and objects in the environment. The scanner allows the user to select objects of interest in the environment, such as tagged documents, other PDAs, printers, book covers, telephones, merchandise, room nameplates, appropriately captioned photographs on the wall, etc., by scanning their IDs. Ideally, the user should be able to scan objects at a distance. In the context of Ambient Combination, scanning an object to select it is referred to as 'zapping'. The user has complete control of what is zapped and what is not zapped. Ideally, where the possibility of one-handed operation is an advantage, the scanner should form an integral part of the PDA, for example as a PCMCIA module plugging into the top of the PDA. In our current prototypes we are using separate 30 foot range barcode readers, but other technologies such as infrared scanners or RFID radio frequency id-tags would be similarly applicable. The PDAs are wireless networked so that they can access object directories, retrieve representations of objects stored elsewhere, and consult DC brokers, all of which will be explained explain later. When initially switched on, the PDA gives access, via applications and hierarchical menus etc, to all of the commands available in the system, just as in any typical commercially available PDA - as shown in Fig 1a.[2]

3.2 Scope of scenarios

These scenarios were devised as exercises to help us to understand and refine the interaction techniques and infrastructure needed to make AC work: their role is not to highlight domains and situations in which AC is especially valuable. For characterisations of situations and domains in which the ability of DC to reduce search is particularly beneficial, see section 7.

---

[2] For enacting some scenarios, laptops were used to simulate PDAs - see section 8.

Scenario 1.1 (Document/Printer)

Anne walks into Ben's office one evening to use his coffee machine. While Anne leaves with her coffee in one hand, Ben mentions that he is reading a paper document that might interest her. Anne wants a printed copy to read later. To save the inconvenience, time and trouble of searching for the document on a server (Ben can't remember the pathname or what the file is called), Anne one handedly uses her PDA to 'zap' the bar code on the document and then 'zaps' the printer in Ben's office. One suggested action is presented on the PDA (figure 1c). Anne presses the 'Do it' button on her PDA to accept this action. The document is printed on the printer. The suggested action was. *Print 'TMA06' to Ben's Printer* (see figure 1c) .

To understand this interaction, it is instructive to consider the state of the interface on Anne's PDA at three successive stages, as shown by figures 1a, 1b and 1c.

a) Figure 1a (*no items zapped)* shows the options available before any items were zapped. The PDA simply displays a choice of application programs and some immediate actions - precisely what a current PDA would offer if direct combination were not available.

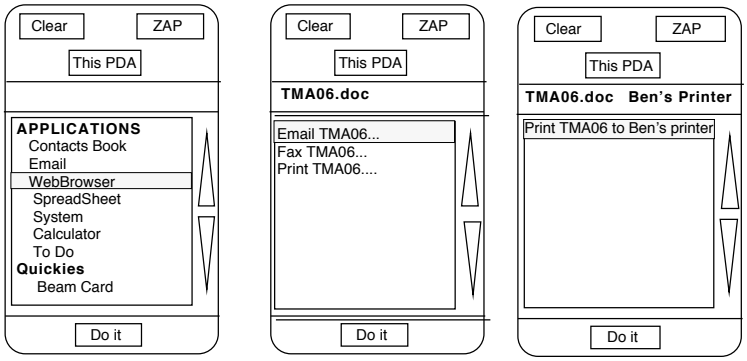

Figure 1a Figure 1b Figure 1c

Figure 1a shows *no items zapped*; Figure 1b shows *document alone zapped;* Figure 1c shows *document and printer zapped.*

b) Figure 1b (*document alone zapped)* shows the options available after the paper document alone has been zapped, with a default highlighted. The options are

Email 'TMA06.doc' ……

Fax ' TMA06.doc' …….

Print ' TMA06.doc'. …….

These actions are those that are relevant when all that is known is that the document has been selected. Any of these options could be chosen by the user straight away and an argument filled in manually or by more zapping. There is nothing new about such

a facility - its equivalent on the desktop is called a "contextual menu". In this situation, the lack of novelty is a virtue: it means that, whenever only a single item is zapped, DC does not get in the way of the conventional, non-DC way of doing things. If the user decides that the wrong item was zapped, the cancel button may be pressed to unselect that object, in which case the situation reverts to figure 1a.
c) Figure 1c *(Both items zapped)*. This option has already been described. Zapping both the document and printer icon greatly reduces the search space of applicable actions, avoiding the need to drill down or otherwise search on the PDA or desktop machine. Note that the options do not depend on the document or printer alone - they depend on the ordered pair of object types that have been zapped - and would in general be different for a different pair of objects. Typically there is still more than one option presented - in that respect, this first pairwise interaction is unusual. If the presented action(s) did not seem appropriate to Anne or if something had been zapped by mistake, she need merely press *cancel* to get back to the state where only a single item, the document, was selected. At this point, the options available would revert to those in figure 1b. Anne might choose to select some other zappable object (e.g. one of the wallcons (see below) to initiate another way of achieving her goal. Or Anne might press *cancel* again to revert back to the situation where nothing is selected.

Scenario 1.2 (Document/Person) Anne might want to have the document emailed to her rather than printed. Anne can achieve this using DC in many ways. For example, Anne could use her PDA to zap the document and then zap her own id badge, if she is wearing one. Her PDA might then offer the following options, with the first as the default.

Email 'Report42.doc' to self
Fax 'Report42.doc' to self

Scenario 1.3 ("This PDA") If Anne had not been wearing a badge, then she could have scanned the document and pressed "this PDA" button marked on her PDA instead. Pressing this button is equivalent to the PDA zapping itself. Amongst other roles, PDAs often act as proxies or representatives of their owner. For this reason, the same options are presented in this case as when Anne scanned her own badge.

Scenario 1.4 (Wallcons)
As a trivial variation of scenario 1.1, Ben might not have a printer in his room, but might have 'wallcons' on his wall (physical or projected captioned wall icons) representing various commonly used entities or resources, such as colleagues, departments, companies, projects, printers, locations, etc. Anne uses her PDA to zap

the bar code on the document and then zaps the printer wallcon on Ben's office wall. The following two suggested actions are presented on the PDA.

Print Document 'Report42.doc' to Room 308 printer

Print Document 'Report42.doc' to meeting room printer

Anne presses the 'Do it' button on her PDA to accept the default.

Figure 2 shows a version of this scenario as handled by the current prototype implementation.

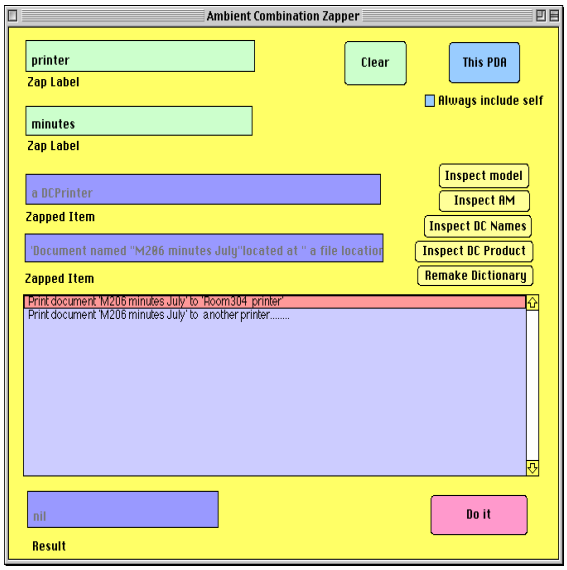

Figure 2 - The implemented prototype being used to enact version of scenario 1.4 , in which a document interacts with a printing facilities wallcon.

4 Principles of Ambient Combination

We are now in a position to state the principles of Ambient Combination, adapted from the DC principles, to deal with mobile and ubiquitous interaction, and to comment on various issues (See Box). We will comment on the three principles of DC in turn.

4.1 Visibility. Ambient Combination requires that as far as possible, every object of interest in a given environment (and in a user's computer) should be visible and capable of zapping (i.e. selection and identification using a pointing device). Issues of privacy, dignity, permission, safety and security must be met, and we will return to these. The principle of *visibility* is borrowed from Direct Manipulation. However, many desktop interfaces that pay lip service to Direct Manipulation often fail to make

objects of interest visible. In the area of mobile and ubiquitous computing, visibility needs special re-emphasis - especially, for example, where objects of interest may initiate uninvited context-aware interactions (Erickson, 2000).

4.2 Pairwise Interaction (more generally n-fold interaction). In many existing systems, commands may be constructed by the user only in a restricted, unidirectional manner. The initial noun followed by the verb must be selected before any arguments can be specified. The order of assembling the command would not matter if it were not for the fact that these restrictions have significant implications for the search

Principles of visibility and pairwise interaction
Every object of interest, *both virtual and in the environment* should, *subject to social, organisational and legal issues including privacy,ownership and* ***security,*** be
(i) visible (or perceptible)
(ii) capable of a **range** of **useful** interactions with **any other** object of interest.
The available interactions between pairs of objects should be:
• diverse,
• tailored to each pair of object types.
Principle of subsumption
DC interaction styles should be implemented in such a way that they are immediately available, but do not obstruct immediate access to existing interaction styles.

strategies available to the user, who may have a clear intended action but cannot recall or is ignorant of the appropriate verb to achieve it. The user may not know the name of a command, how it is categorised, whether it exists, or where to find it. Once the user has selected the initial object of interest, this does not always greatly constrain the space of applicable commands, so the user may still have a large space of verbs to search. If the user can indicate in advance *two or more* of the interaction objects involved in an intended command this greatly *constrains the search space of possible interactions*. This allows the system to present a space of focused relevant commands. If the choice of focused commands offered is inappropriate, the user may deselect items or select other items of interest until the system makes suggestions that are more appropriate.

The requirement that **every** object of interest be capable of diverse interactions with **any other** object of interest is a heuristic to encourage designers to seek meaningful pairwise interactions (and to endow objects of interest with suitable operations) rather than an absolute universal requirement. The reference to *virtual* objects refers to objects of interest represented in a computer. When using Direct Combination, it should be possible to mix both virtual and real objects in a single interaction.

4.3 Subsumption. Subsumption ensures that users are never worse off with a system that includes DC than with a system that excludes it. For example, in the scenarios, all of the traditional ways of using a PDA are available when nothing is selected: hence users can still make use of the familiar *verb, arguments* pattern or the *noun, verb, arguments* pattern. DC may be viewed as, and is easily implemented as, *subsuming* these patterns of interaction. The principle of subsumption is a common sense precaution, as it is not always convenient to use pairwise interaction. DC does not impose anything: it simply provides greater freedom for users. In our prototype, the DC command processor only processes commands in the pattern *noun* and *noun noun*. Conventional applications are relied on to process commands in other patterns. A DC command processor that fully implemented subsumption would process commands using all of these patterns.

5 Context-Aware Interaction

The purpose of the next set of scenarios is to illustrate the way in which Direct Combination can afford context-aware interactions. Note that each of the examples below involve Anne's PDA (which does the zapping in each case) as one of the zapped objects (e.g. via the 'this PDA' button).

Scenario 5.1

Anne meets a research colleague Fred in the corridor. Anne zaps Fred's PDA with her PDA. The options offered are as follows.

Email 'Research draft' to fred
Email 'Research notes' to fred
Search folder 'ACDC 'for documents to email to fred
Arrange meeting with Fred

Scenario 5.2

Anne meets a teaching colleague Tony in the corridor Anne zaps Tony's PDA with her PDA. The options offered are as follows.

Email 'TMA04.doc' to tony
Email 'TMA tutor notes' to tony
Search folder 'TMA04 2002' for documents to email to tony
Arrange meeting with Tony

Scenario 5.3

Anne goes to a course team meeting and sees Rob with whom she collaborates both on teaching and research. Anne zaps Rob's PDA with her PDA. The options offered are as follows (see Figure 3).

Email 'LearningBook4' to rob
Email 'LearningBook3' to rob
Search folder 'Learning Book revisions' for documents to email to rob
Arrange meeting with Rob

In each scenario, different interactions are offered, although the ordered pair of object types is the same each time - two PDAs. This reflects the fact that for some ordered pairs of objects types (e.g. PDX x PDA), the options offered may depend on the state of the objects as well as on their types. In scenarios 5.1 and 5.2 the key state difference is the identity of the owners of the zapped PDAs, and the nature of their work relationships with Anne, as found by the Object Broker when it assembles an object model for the zapped objects. In scenario 5.3, the automatically sensed time and location of the interaction would be used by the Object Broker to make heuristic inferences about the most relevant options to offer. (Note: in our prototype, the sensing of contextual information in case 5.3 is currently simulated, and the coding used to make inferences from this contextual information is fairly ad-hoc and unsophisticated.)

In general, to make the most of Ambient Combination , object models that describe the environment need to be maintained. There are many ways in which this could be done. In the discussion of the scenarios below we outline one possible way in which the maintenance of those parts of these models that deal with personal information could be performed. (For our prototypes, we hand-coded this information.) For example, in scenario 5.1, we assume that Anne, at some time prior to the scenario, originally chose to exchange cards with Fred via beaming between PDAs. We will that at that time, the id-tag of Fred's PDA was automatically included together with the normal kinds of contact information about Fred that was transferred to Anne's contacts book. This allows Anne's PDA in principle to recognise Fred's PDA subsequently and associate it with Fred.

Further, we can imagine that Anne uses drag and drop on a hypothetical DC desktop to associate *projects* and activities with related *documents* or with folders holding related documents. Similarly we can imagine that in the address book on her PDA or on her hypothetical DC desktop, Anne uses drag and drop to associate *colleagues* with *projects*. In such a manner, an individual's diary, address book and file system could be used in a fairly straightforward way to maintain private elements of the object model used by that individual.

Of course, not everyone would wish to organise their address books or desktops in such ways. However,.this does illustrate one way in which private object models could be maintained with little effort.

For the purposes of the scenarios, Anne uses a private Direct Combination broker, as we will outline below. The private DC broker could live on Anne's PDA or could live on her desktop machine or on a server.

Anne's DC broker contains a well-factored object model of objects typically encountered in Anne's environment. In real life, we might imagine that some of the classes in this model (for example a standard set of base classes) would be common to more or less all users, and so versions of this might be supplied as standard by manufacturers, or otherwise made freely available. Other parts of the model (for example other, more specialised classes) might be provided by Anne's professional association, or by other special interest groups to which she subscribes. We are assuming that, in addition, Anne chooses to give her own object broker access to information about what colleagues are associated with what activities etc, as outlined above. (In our prototype all of the class hierarchy of objects of interest was hand crafted.)

Now that we have indicated something about the general nature of the object model required for the scenarios, and how parts of it might be maintained, we can now recap the mechanisms of the interactions in scenario 5.1.

In scenario 5.1, when Anne zaps Fred's PDA and crosses it with her PDA (i.e. both items are zapped), the DC broker attempts to build up a little object model of the zapped objects. Anne's broker does not have permission to access Fred's PDA, which is private, but from its scannable id-tag, the broker can find out the PDA's owner, via existing recorded associations, obtained as outlined above. In a similar way, the Broker can also retrieve any relevant associations recorded between Anne and the owner. Again, as noted above, these might have been recorded via Anne's address book or via drag and drop on her desktop. In scenario 5.1, as already noted, the recorded association between Anne and Fred that is retrieved concerns a particular project, and this has associated documents and folders. Consequently the most up to date documents relevant to the research project are offered to be sent to Fred. Note that a broker belonging to another colleague might have offered quite different options in similar circumstances, since there is no reason why DC brokers should not be be personalised in a variety of ways, e.g. according to professional function or personal preference.

Scenario 5.2 is very similar to scenario 5.1, except that the relevant relationship between Anne and Tony is that they are on the same teaching team. Consequently, the

most up to date documents relevant to the course team are offered to be emailed to Tony.

According to some definitions, the two examples considered so far have not involved contextual information, since the 'contextual' information merely regarded inference of relevant relationships based on looking up the owner of the zapped object. The next example, by contrast involves location information that could be automatically sensed.

In Scenario 5.3, Rob has two relevant relationships with Anne - he shares a course team with Anne, but is also a collaborator on a research project. One reasonable approach for an Object Broker in computing options to offer might be to simply combine the options thrown up by the two contrasting relationships (and some brokers might reasonably do that). Instead, we have used the example to illustrate how automatically sensed contextual information could be applied within the DC framework.

In particular, information about location and time is used to decide heuristically which relationship should take precedent in deciding options to offer to the user. Note: the word 'heuristically' is not intended to imply that any sophisticated inferencing is used - the 'inferencing' is in fact rudimentary. We are simply emphasising that the system is guessing, and could guess wrongly. Also, recall that in our prototype we are currently simulating the sensing of location. For these reasons, scenario 5.3 should be treated as hypothetical.

In Scenario 5.3, Anne's location is sensed to be in a meeting room, and the time of day is found to match a meeting previously booked in Anne's diary (which she associated at the time by drag and drop with the course team in question). Furthermore, the booking diary associated with the meeting room independently shows a meeting booked with a course team of which Anne is a member. Until Anne actively zaps one or more objects, these automatically sensed context objects (the location and the meeting) do nothing in the DC system. Let us now consider what happens when Anne zaps her PDA and Rob's PDA.

In our early attempts to work out how context-aware DC systems should work, we explored a scheme whereby automatically sensed context objects should be included in the combination (though marked as arising from context, as opposed to deliberate zapping) along with explicitly selected objects. This approach has a pleasing conceptual uniformity and consistency. However, it is currently technically demanding, and experience has led us at the present time to prefer at present a system where automatically sensed context objects are used instead during combination to influence heuristically the filling out of the mini object model of explicitly zapped objects. In the current case, the context parser heuristically prunes part of the object

model giving preference to options focusing on the teaching relationship, as opposed to the research relationship (figure 3).

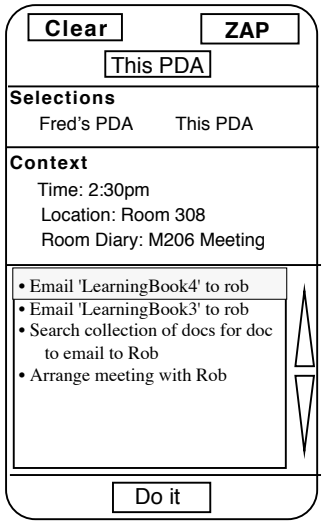

Figure 3. A hypothetical context-aware interaction.

Scenario 5.4

Anne meets a stranger and Anne zaps the visitors PDA with her PDA. The following option is offered.

Beam card to strange PDA

This scenario is offered as the base case. No previous relationship is recorded with the stranger, so only options appropriate for strangers are offered. However, when cards exchanged, the id-tags on the PDAs may be recorded, so that if either party goes on to annotate the other's card at a later point with discovered interests or roles in common, more tailored options might be offered if they met again.

Context aware interactions

Let us now review some issues arising from last four scenarios. Ambient Combination need not involve context-aware interactions at all. However, preliminary work suggests that when contextual information is used, Ambient Combination can bring distinctive new elements to context-aware interaction, as follows.

Recall first of all some technical detail. As already noted, one option for dealing with context-awareness under AC is to call for items of context, such as the time and location, to be considered on a par with deliberately zapped objects, such as a PDA or a printer. This approach is attractive, but at the moment we are pursuing a contrasting approach where context objects are treated in a different way from explicitly zapped

objects. Context objects are used heuristically to: specialise classes of zapped objects; fill in attribute values and roles; propose missing objects; clarify relationships between zapped objects; and generally to heuristically help instantiate the object model that results whenever one of more objects are zapped. In other words, in our current prototypes, we propose to use context-awareness heuristically to fill in detail about objects to be combined. In summary, such 'filling in' may involve specialising the class of an object, providing a missing object, or filling in details of state. (In the prototype this is done in a rudimentary manner).

When are now in a position to explain the ways in which this usage of contextual information appears a little different from the way in which context is generally applied in mobile HCI. Firstly, when a user selects two or more objects in the environment, this can be used to narrow down the set of intended actions, affording a substantial reduction in the search space. By applying contextual information at this late, knowledge-rich stage information about the class, state and mutual role of objects can typically be refined heuristically, reducing the search space even further. This use of contextual information at a knowledge rich stage contrasts with many existing context-aware systems, which guess desired actions based on typically much less constraining evidence such as automatically sensed information alone, or automatic interpretations of users actions (Erickson, 2002).

A second contrast with conventional context-aware systems is that, as is always the case in DC, the choice of action is left to the user, whether contextual information is used or not - actions are not taken automatically. Of course, existing context-aware systems could be made similarly conservative. However, to the extent that AC reduces search and renders many interactions more convenient for users to specify at least partially, it can lessen the need for the system to have to guess blindly when applying contextual information.

Finally, a third difference is that under AC, automatically sensed context objects, where used, are always made visible to the user, who may opt to overrule the use of any particular item of context for a given interaction, or in general (e.g. "ignore the diary until further notice"). More generally, because the process of applying context is heuristic, the user always has the option of switching off the use of context in general, or the switching off of particular kinds of context (e.g. location sensing, activity inference based on personal or departmental calendar, use of address book etc) either as a matter of preference until further notice or for given interaction .

The way in which information about context is typically *detected* is no different in AC from other uses of context, (though as we have argued above, the way in which the context is *applied* is different in significant respects).

6 Architecture

For AC to be deployed in a given environment, various architectural elements need to be present. AC can be implemented in its simpler forms relatively simply. The environment needs to be made scannable in some way. We use barcode readers, including a thirty-foot range hand-held scanner, and barcodes, as stand-ins for a wide range of possible alternative technologies. The user needs a feedback device such as a PDA screen. A DC broker is needed (a system which, given references to two objects, computes the options to offer the user). This may be held locally (e.g. on PDAs and other devices) or on a central server. Distributed architectures are also possible. A DC Broker is a collection of objects (sometimes created on demand as needed) representing scanned objects and informed by an appropriate, well-factored, class hierarchy. In many cases, only the class of a scanned object need be known, not its state. However, in some cases, state may needed to help infer relevant operations (e.g. when context is used). Operators need to be distributed over object type pairs (preferably using abstract types) - see Holland and Oppenheim (1999) for various design approaches. Unless a distributed architecture is chosen, the design of an AC Broker can be similar to that of a desktop DC Broker (Holland and Oppenheim 1999).

6.1 Security

In a fully deployed AC system, scanned objects might not reveal their identity to all enquirers; objects might not reveal their state; and diverse networked DC Brokers might be available for different specialised purposes to those with the right credentials. Hence, much like any other interactions on computers, some AC interactions are likely to be freely available to all, others available only to those who have paid subscriptions, or who have the correct passwords. Similarly, some kinds of interaction might be freely available to all within particular buildings such as railway stations, airports, libraries or supermarkets. Yet other interactions might be available only when initiated from accredited pointing devices, themselves password protected; such arrangements might apply to schools, universities, places of work, policemen, emergency workers, and maintenance workers as appropriate.

7 When is DC useful?

Because of the principle of subsumption, DC does not have to be useful all of the time in order to be beneficial. Even if it only helps some users with some problematic interactions some of the time, then DC can be useful, since it is designed not to get in the way. But it would be useful to characterise the kinds of situations when DC saves work. Empirical work is needed to answer this question, but since the DC hinges on reduction of search, some a priori considerations can help characterise such situations. In this section we consider a priori considerations of this sort. Domains can be

expected to work well with DC given sufficient variety of objects, sufficient variety of operations, and provided that the operations are distributed between object type pairs. For AC (over and above Desktop DC) to be applicable, a scannable environment is needed where the objects of interest are visible in the environment.

Given a suitable domain, users could be expected to benefit from DC in any of the following situations: the domain is unfamiliar; there are too many commands to search quickly; the user is focussed on the environment; minimal attention situations; feedback bandwidth is limited; or the task focus is on a combination of objects. DC is particularly useful for all kinds of data translation and interoperability. As noted earlier, much work with mobile and ubiquitous devices involves ad-hoc tasks with novel combinations of unfamiliar resources - often in situations of minimal attention. AC appears to be well suited to such situations.

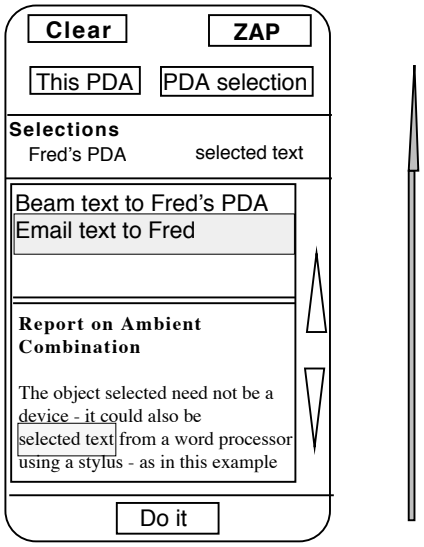

Figure 4 A hypothetical interaction between physical and virtual objects

Finally, note that although the scenarios in this paper have focused on interactions between physical objects, this is not a limitation of AC. AC can just as easily be applied to interactions between physical and virtual objects in any combination. The only extra requirement is that user interface must afford the means to select virtual objects as well as to zap physical objects. See figure 4 for a hypothetical PDA interface of this kind. The user may select in the normal way, using a stylus, any item that is displayed on the PDA (e.g. selected text in a word processor). If the user then

presses the "PDA Selection" button, this zaps the selected item (in this case selected text) for Direct Combination with some other object.

8 Prototypes and their limitations

The prototypes reported here were designed for purposes of proof of concept and for refining the DC mechanism and the example interaction styles we have used. The architecture used in the prototype is not scaleable to n-fold combinations (i.e. when three or more objects are deliberately zapped per interaction) The domain model for our prototype DC Broker was kept minimal. Some of the combinations whose result should vary depending on the state of one or more zapped objects were dealt with in the prototype in ad-hoc ways that we would now avoid by using the object-oriented Design Pattern 'State' (Gamma et al, 1994). We did not connect up the broker to the relevant services (e.g. print queue, etc) - the user was presented with textual stubs. Our prototypes are designed to work on Casio PDAs via front ends coded in Squeak Smalltalk communicating by wireless networking to a Direct Combination broker coded in Smalltalk running on a G3 Macintosh laptop. However, the ambulant demos reported here were carried out with laptops simulating PDAs. Each laptop used a locally held copy of the DC broker coded in VisualWorks Smalltalk. The identities of zapped objects were looked up via a local directory. Tagging of objects in the environment was achieved using barcodes and USB hand-held bar-code scanners, including a long-range scanner, which can read bar-codes at a distance of 30 feet. Automatic sensing of different locations was simulated. The prototype broker followed the subsumption principle and could offer relevant actions whether 0, 1 or 2 objects were zapped.

9 Related work

Using pointing devices to transfer information between devices (typically computers) is not new. For example, Pick-and-Drop (Rekimoto 1997 1998) is a pioneering extension of Drag & Drop used to copy data between multiple devices via passive pens with ids. Pick and Drop can be viewed as a natural extension of Drag & Drop to mobile and ubiquitous computing. The principle of Ambient Combination may be seen as an extension of Pick and Drop. AC offers potentially greater flexibility and expressiveness of interaction. The InfoStick (Kohtake et al, 1999) is an interaction device for inter-appliance computing. It may be used to pick up and store information items from a variety of devices and then move them to other devices. The InfoStick may be viewed as offering a limited special case of Ambient Combination where the only available operations are *get* and *put*.

DataTiles (Rekimoto et al, 2001) is an attractive tangible computing system using 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 to make this continuous or discrete in nature. DataTiles, like many tangible computing system, has the potential to be given greater flexibility and power, without loss of elegance, by applying the Ambient Combination principle. For example: AC options could be visibly offered when two tiles were placed together or connected by pen. Alternatively, subsumption would be better satisfied if AC interactions were invoked via a single additional pen gesture to invoke Direct Combination between two tiles. Alternatively a special magic lens tile could be laid on top of inter-tile borders to reveal and allow selection of available interactions.

10 Conclusions

Ambient Combination (AC) is the adaption, extension and application of Direct Combination (DC) to Mobile, Ubiquitous, Tangible and Mixed Reality Computing. In many situations, Direct Combination can reduce the degree of search required by users to carry out actions that they do not know how to perform quickly. Examples have been given of AC interaction techniques. AC is useful where display real estate is limited or input devices are inconvenient. It is useful in minimal attention situations, and situations where users need to make unfamiliar devices work with other devices. Even where individual devices have well-designed user interfaces, this kind of operation can be difficult. AC lets the user construct tightly focused *combinatorial implosions* to combat the combinatorial explosions inherent in these and other circumstances. Ambient Combination allows the user to exploit objects in the environment to narrow down the range of interactions. AC is not a replacement for other kinds of user interaction; it is a complement: standard interaction patterns may be viewed as special cases. AC stays out of the way of users when it is not useful. When pairwise combination is applicable, it reduces the need for memorisation and interface navigation. AC allows users to employ recognition as opposed to recall. Ambient Combination may be viewed as a new way of distributing the user interface in the environment. Ambient Combination seems to offer a new approach to dealing with context-aware interactions. The strength of AC arises from the twin exploitation of user's perceptual, physical and spatial situatedness in the environment, and the organisation inherent in object and inheritance hierarchies. Almost all of the *elements* of Ambient Combination exist in fragmented or partial form in other systems: what is new is the simple uniform framework and approach that DC affords for dealing with challenging interactions, and simple abstract architectures to make this possible. We argue that AC has a useful role to play in any future framework for mobile and ubiquitous HCI.

Acknowledgements
Thanks to: Alistair Edwards, Claudia Eckert, Martin Stacey, Randy Smith, Bill Gaver, Benedict Heal. A shorter and adapted version of this paper was submitted to Mobile HCI 2002.

References

Binsted, Kim (2000) Sufficiently Advanced Technology: Using Magic to control the world. In Extended Abstracts of the ACM Conference on Human Factors and Computing Systems CHI2000, pp 205-206 ISBN: 1-58113-216-6.

Erickson, Thomas (2002) Some Problems with the Notion of Context-Aware Computing. Communications of the ACM Feb 2002/Vol 45 No.2 pp102-104.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1994) Design Patterns. Addison Wesley, New York.

Holland, S. and Oppenheim, D. (1999) Direct Combination. In Proceedings of the ACM Conference on Human Factors and Computing Systems CHI 99, Editors: Marion Williams, Mark Altom, Kate Ehrlich, William Newman, pp262-269. ACM Press/Addison Wesley, New York, ISBN: 0201485591.

Kohtake, N., Rekimoto, J. and Anzai, Y. InfoStick: an interaction device for inter-appliance computing, in workshop on handheld and Ubiquitous Computing (HUC '99) 1999.

J. Pascoe, N. Ryan, and D. Morse, "Using While Moving: HCI Issues in Fieldwork Environments," ACM Transactions on Computer Human Interaction, vol. 7, pp. 417-437, 2000.

Rekimoto, J., Pick and Drop: A Direct Manipulation technique for multiple computer environments, In proceeedings of UIST '87 p.31-39

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

M. Weiser, "The Computer For the 21st-Century," Scientific American, vol. 265, pp. 66-75, 1991.