Department of Computer Science and Software Engineering
University of Canterbury

March 2011

Bag-and-Dump: Design and Evaluation of a User Interface for manipulating items across multiple contexts.

A thesis submitted in partial fulfilment of the requirements
for the Degree of Master of Science in Computer Science
in the University of Canterbury by

Dominic Winkler

Supervisor: Professor Dr. Andy Cockburn
Associate Supervisor: Associate Professor Dr. Tim Bell

To my family for their endless love and support.

Abstract

The copy-and-paste paradigm is a fundamental operation in graphical user interfaces. However, existing copy-and-paste techniques have limitations, in particular in terms of efficiency and robustness against interruptions. This thesis is focusing on improving the user interface used to copy-and-paste objects across different contexts, such as a series folders. To improve this fundamental operation, a new copy-and-paste technique, called **Bag-and-Dump**, is proposed, implemented and evaluated. Bag-and-Dump aims to substantially reduce mouse movement by allowing the user to gather up ('bag') source data across different folders before 'dumping' the whole load at the destination. Additionally, Bag-and-Dump provides constant visual feedback in the form of a bag-like semantic cursor to increase robustness against interruptions. Bag-and-Dump was evaluated against two standard copy-and-paste techniques (Keyboard Shortcuts and Drag-and-Drop) under a different number of contexts (folders) and with and without interruptions. Results from the experiment not only showed that Bag-and-Dump indeed significantly reduces mouse movement, it also confirmed that Bag-and-Dump was 9% faster than Keyboard Shortcuts, one of the most popular copy-paste techniques among "expert users".

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank Andy (my supervisor), Hugo-and-James-and-Tommy (my proof-reader gang), my parents and sister, Jessie, and all the other people (whose names I might not remember but have helped me in some aspect) for their support and love throughout this Masters.

CHAPTER 1

Introduction

Selecting multiple objects and then manipulating them is a common interaction used by millions of computer users several times a day. The copy-and-paste paradigm is one example of this generic select-manipulation interaction. It enables the replication of a part of a document in the same or another document and provides the ability to re-organise files in the computer's filesystem. The copy-and-paste paradigm, which became available in early systems such as Sketchpad or Xerox Star, is still considered to be one of the fundamental services provided by graphical user interfaces today.

Modern operating systems usually offer three different standard techniques for copy-and-paste: *Keyboard Shortcuts*, *Menus* and *Drag-and-Drop*. These techniques follow a four-step-procedure: select, copy, specify, paste; whereas all three use the same methods to select objects and specify the destination, they use different means to invoke the copy-paste commands. Even though copy-and-pasting plays an essential role in the creativity and productivity of today's computer users, these interaction techniques reveal limitations, in particular in terms of efficiency and robustness against interruptions.

Imagine, that you want to compile a slideshow of the best photos of the last five years. All of your photos are organised in a hierarchical file structure, ordered by their year. Your task involves choosing photos from various folders and organising them into one shared destination folder. For this you require the copy-and-paste paradigm. Regardless of which technique you decide to use, to copy-and-paste all desired photos you have to go back and forth between the source folders of each year and the destination folder, giving a path of *2n-1* nodes, where *n* is the number of different source folders. This is necessary because selections do not carry across multiple selection contexts (such as folders). But can this long and potentially frustrating path be shortened? What if you could select the best photos of each year in one go and then copy-and-paste all of them at the same time into the destination folder. This would reduce your mouse movement to a path of *n* nodes.

Long iteration paths reduce efficiency, but copy-and-paste is also susceptible to errors due to interruptions. For example, imagine you are in the middle of compiling the above slideshow, when the phone rings. You answer, but then after the call you realise you have forgotten what you were doing. Your workflow has been interrupted, a common concurrence in today's working life. To complete the interrupted task you have to perform additional operations re-orientating yourself to the task. This is necessary because existing standard copy-and-paste techniques do not provide a method for identifying relevant information to complete the disrupted task. But can these additional operations also be avoided? What if additional visual feedback was provided, informing you of the relevant information to complete the disrupted task. This would spare you from performing these additional operations.

This research focuses on improving the user interface used to copy-and-paste object(s) across multiple contexts, such as a series of folders. To improve this fundamental operation, a new copy-and-paste technique, called **Bag-and-Dump**, is proposed, implemented and evaluated. Bag-and-Dump reduces the path length to *n* by allowing the user to gather up ('bag') source data across different folders before 'dumping' the whole load at the destination. To be able to do this, Bag-and-Dump introduces a new selection mode that enables the user to selected items from different contexts in one go. The selected items are displayed in a 'bag'-like semantic cursor that provides constant visual feedback to inform the user of the system's current copy-paste status. In case of interruptions, the user can regain relevant information to complete the disrupted copy-paste task from the displayed 'bag'.

To evaluate this new copy-and-paste technique, Bag-and-Dump is compared to two standard copy-and-paste techniques (Keyboard Shortcuts and Drag-and-Drop) under a different number of contexts with and without interruptions. Results from this experiment show that Bag-and-Dump significantly reduces mouse movement. Bag-and-Dump was also the preferred technique and was 9% faster than Keyboard Shortcuts, one of the most popular copy-paste techniques among "expert users".

Chapter 2 of this thesis discusses different underlying human factors that impact the capabilities and limitations when executing copy-paste operations. The same Chapter also reviews the standard copy-and-paste techniques as well as extensions of them and analyses the relative merits of these techniques. Chapter 3 describes Bag-and-Dump, an improvement designed to overcome limitations of existing copy-and-paste techniques especially when copy-and-pasting across multiple contexts. Chapter 4 details an experiment and its results comparing Bag-and-Dump to two standard copy-paste techniques as well as reports findings of participants' copy-paste practices. The experiment's results and implications are discussed in detail in Chapter 5 and solutions addressing problems identified in the experiment are proposed. Finally, Chapter 6 summarises the findings of this research.

CHAPTER 2

Related Work

This chapter presents previous work related to *copy-and-paste*. Section 2.1 introduces the general *copy-and-paste paradigm* in more detail. Then existing standard techniques to perform copy-and-paste operations are presented in Section 2.2. Section 2.3 discusses underlying human factors that impact the human capabilities and limitations (physically as well as mentally) when executing copy-and-paste operations. Section 2.4 presents existing extensions of the standard copy-and-paste techniques and discusses their strengths and weaknesses. The final section compares all previously mentioned techniques.

2.1 Copy-and-Paste Paradigm

Copy-and-paste is a basic interaction that most of us probably learnt in kindergarten, using scissors and tubs of glue to cut out existing snippets and glue them together to create a new “piece of art” [Renard, 1999]. As adults, however, *copy-and-paste* became words that we associated with the process of gathering and organising data from various sources. This ability to manipulate data is one of the most practical and essential tools for today's computer users [Shneiderman, 2000].

With the introduction of Sketchpad [Sutherland, 1963] and later on Xerox Star [Johnson, Roberts, Verplank, Smith, Irby, Beard, and Mackey, 1989], the concept of data exchange between multiple documents or applications has become a fundamental operation in graphical user interfaces. Both *cut-and-paste* and *copy-and-paste* follow the same four-step action sequence [Faure, Chapuis, and Roussel, 2009]: (1) Selecting one or more object(s) by using an input device (e.g. the mouse), (2) activate either the copy or cut command, (3) specify the destination (which could be in the same window or another), and (4) finally, activate the paste command.

Although this thesis mainly focuses on copy-and-paste, the only difference between copy-and-paste and cut-and-paste is that after invoking the cut mode the selected objects are copied and then deleted from their original location.

2.2 Standard Copy-and-Paste Techniques

There are three different standard techniques that support this four-step action sequence: *Keyboard Shortcuts*, (pop-up) *Menus* and *Drag-and-Drop*.

The first step of the action sequence of all three techniques is similar – selecting one or multiple items. From that point onwards techniques differ in the way users execute auxiliary tasks as well as invoke the copy-paste commands.

2.2.1 Selection

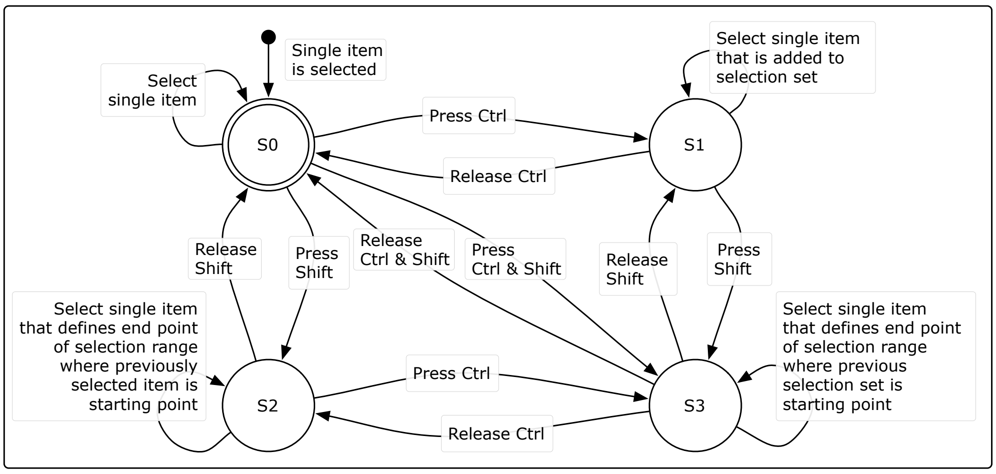

Single item is selected
Select single item
S0
Press Ctrl
Release Ctrl
S1
Select single item that is added to selection set
Release Shift
Press Shift
Release Ctrl & Shift
Press Ctrl & Shift
Release Shift
Press Shift
Select single item that defines end point of selection range where previously selected item is starting point
S2
Press Ctrl
Release Ctrl
S3
Select single item that defines end point of selection range where previous selection set is starting point

Figure 2.1: State diagram of standard selection modes

Regardless of the technique used, selecting the target item(s) is the first step of the action sequence. By using an input device, e.g. the mouse, the user moves the cursor onto the item to be copied. Then by clicking on the left mouse button, the desired item gets highlighted and selected. This is the single item selection mode represented in State S0 in Figure 2.1. To be able to select more objects, the user needs to change the selection mode, which is done via modifier keys, e.g. Shift and Ctrl when using Microsoft Windows. Pressing Ctrl, as shown in State S1 of Figure 2.1, enables the user to select further single items, adding one at a time to the selection set. When pressing Shift the user can add a range of items to the selection set by first selecting the start item of the range and then the last object (State S2). By pressing Shift as well as Ctrl (State S3) the user can add a range of items to a previously selected selection set. The previously selected items define the starting point while the last item defines the end point of the range. Hence the user can choose between four different selection modes:

1. **State S1** – Single item selection mode;
2. **State S2** – Multiple single items selection mode;
3. **State S3** – Range of multiple items selection mode;
4. **State S4** – Additive range of multiple items selection mode.

2.2.2 Auxiliary Tasks

Auxiliary tasks, in particular in the form of window management tasks, are becoming more and more important due to the increasing number of windows users have open [Hutchings, Smith, Meyers, Czerwinski, and Robertson, 2004]. Since the average number of simultaneously opened windows increases with available display space and large monitors as well as multi-monitor setups are becoming increasingly popular, the number of open windows will increase [Smith, Baudisch, Robertson, Czerwinski, Meyers, Robbins, and Andrews, 2003]. These open windows are filled with objects to copy and destinations to paste them to, which imposes time-consuming and complex navigation tasks between or inside windows [Chapuis and Roussel, 2007]. This, in turn, leads to an increase in the number of auxiliary tasks users might want to perform between copying and pasting objects. For example, regardless of which technique is being used, a copy-and-paste operation between two different overlapping windows requires users to perform additional window management tasks. These tasks are usually carried out when specifying the destination location for the copied objects during the third step of the action sequence.

The importance and frequency of window switching, also while copy-and-pasting, has encouraged the creation of numerous interfaces [Tak and Cockburn, 2010]. Operation systems offer a variety of interaction techniques such as *Exposé* under Mac OS X and *Alt+Tab* under Windows to support window management. To specifically facilitate window management during copy-and-paste operations, restack and roll [Chapuis and Roussel, 2007] as well as desk pop and stack leafing [Faure et al., 2009] were developed. In general though, regardless of which copy-and-paste technique is being used, users should be able to perform window management and other auxiliary tasks during copy-and-paste operations. Consequently, the way these auxiliary interactions are performed should be independent from the actual copy-and-paste technique.

2.2.3 Command Invocation

2.2.3.1 Keyboard Shortcuts

Sketchpad and the Xerox Star already had specific Delete, Copy and Move keys that in combination with the pointing device could be used to copy-and-paste [Chapuis and Roussel, 2007]. For example, pressing the Copy-key on the Xerox Star attached the previously marked objects to the cursor and another mouse click at the destination pasted the items there. Today's computer systems do not have specific keys for these functions but instead use keyboard shortcuts.

Keyboard shortcuts are an efficient method to invoke common commands [Lane, Napier, Peres, and Sandor, 2005; Tak, 2007]. They expedite common operations by reducing input sequences to a few keystrokes, hence the term 'shortcut'. Most keyboard shortcuts require the user to hold down a modifier key and then press one or even multiple other key(s) simultaneously. In principle, the research community [Nielsen, 1994; Dix, Finlay, Abowd, and Beale, 2003; Lane et al., 2005] agrees that keyboard shortcuts are a way for expert users to increase their level of efficiency while at the same time the learnability for novel users is not compromised. However, as studies show, even expert users often do not use available keyboard shortcuts [Lane et al., 2005; Peres, nad F. P. Tamborello II, Yang, and Paige-Smith, 2005]. This is due to users satisficing in their computer use because other methods are good enough for them [Tak, 2007]. Nevertheless, the copy-and-paste keyboard shortcuts, holding down the modifier key 'Ctrl' ('Control' on Windows) and simultaneously pressing the 'c'-key (Ctrl-C) to copy and Ctrl-V to paste, are well known.

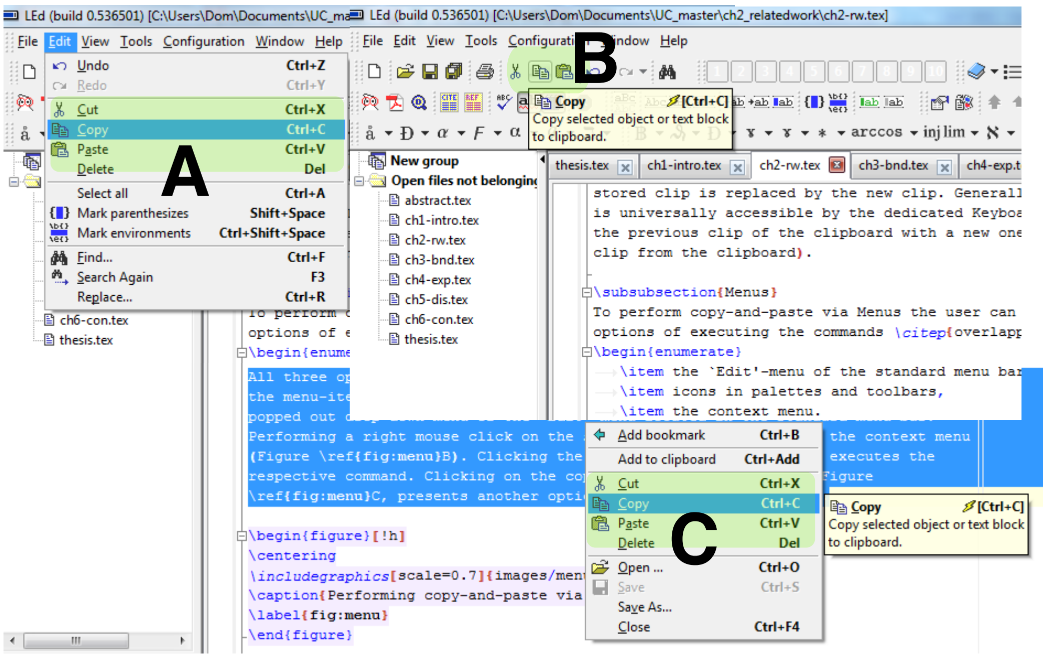
The copy command Keyboard Shortcut Ctrl-C copies the selected objects to the system clipboard. The system clipboard is a temporary storage that is limited to one single clip at a time and invisible at all times. A single clip can consist of one or more objects. Whenever a new clip is copied to the clipboard, the previously stored clip is replaced by the new clip. Generally speaking, the system clipboard is universally accessible by the dedicated Keyboard Shortcuts Ctrl-C (to replace the previous clip of the clipboard with a new one) and Ctrl-V (to recall the latest clip from the clipboard).

2.2.3.2 Menus

To perform copy-and-paste via Menus the user can choose one of three different options of executing the commands [Chapuis and Roussel, 2007]:

1. The 'Edit'-menu of the standard menu bar of an application;
2. icons in palettes and toolbars;
3. the context menu.

All three options require an input device, like the mouse, to move the cursor onto the menu-item or icon to perform the commands. Figure 2.2 A shows the popped out drop down menu of the ‘Edit’-menu located in the standard menu bar. Performing a right mouse click on the selected object(s) pops out the context menu (Figure 2.2 C). Clicking the ‘Copy’-item in these menus executes the respective command. Clicking on the copy icon in a toolbar as in Figure 2.2 B, presents another option to copy selected item(s).



```
stored clip is replaced by the new clip. Generall
is universally accessible by the dedicated Keyboa
the previous clip of the clipboard with a new one
clip from the clipboard).

\subsubsection{Menus}
To perform copy-and-paste via Menus the user can
options of executing the commands \citep{overlapp
\begin{enumerate}
    \item the `Edit'-menu of the standard menu bar
    \item icons in palettes and toolbars,
    \item the context menu.
```

```
To perform c
options of e
\begin{enume
All three op
the menu-ite
popped out
Performing a right mouse click on the                      the context menu
(Figure \ref{fig:menu}B). Clicking the                     executes the
respective command. Clicking on the co                     Figure
\ref{fig:menu}C, presents another opti

\begin{figure}[!h]
\centering
\includegraphics[scale=0.7]{images/men
\caption{Performing copy-and-paste via
\label{fig:menu}
\end{figure}
```

Figure 2.2: Performing copy-and-paste via Menus

As with Keyboard Shortcuts, performing the copy command via menus causes the selected objects to be copied to the hidden system clipboard. To recall the stored clip via menus, the user performs the same actions as when copying but instead of clicking the ‘Copy’-item (s)he clicks ‘Paste’ (Figure 2.2).

Since the user has to move the mouse cursor to reach menu bars, palettes and toolbar icons, these are not in-place techniques. Even the right mouse button activated context menu, which is an in-place menu, requires extra mouse travel to execute a command.

Therefore, in comparison to Keyboard Shortcuts, menu based command operations impose additional mouse travel to activate the commands.

2.2.3.3 Drag-and-Drop

Another way of executing a copy-and-paste operation is the mouse-based drag-and-drop technique that avoids the use of the hidden clipboard all together [Wagner, Curran, and O'Brien, 1995]. The user simply has to press and hold down the left mouse button on one of the previously selected items, drag the mouse pointer (which has the selected object(s) attached to it) to the destination and release the mouse button to drop the object(s) there. Drag-and-drop combines the last three steps of the copy-paste action sequence in one single, continuous, direct manipulation interaction technique [Faure et al., 2009; Gayer, 1989].

This continuous interaction provides clear visual feedback in terms of dragging the object(s), which is reinforced by the tension of the finger holding down the mouse button. “This tension reminds the user that (s)he is in a temporary state and makes syntax and mode errors virtually impossible" [Faure et al., 2009]. Moreover, most systems give additional feedback in the sense that the destination target, e.g. a folder, highlights when the mouse cursor is over it to indicate that the dragged objects can be dropped (Figure 2.3 A & B).

While drag-and-drop's continuous feedback is a great strength it also poses a great disadvantage. The necessity to apply continuous pressure on the mouse button while dragging an object is the technique's most obvious drawback. Holding down a mouse button is not only fatiguing and error-prone but it also seriously limits the interaction techniques usable to execute any auxiliary tasks during a drag [Faure et al., 2009]. To make up for these limitations, users often end up performing a far more complex ‘work-around’ to be able to drag-and-drop object(s) towards a partially or totally hidden window [Dragicevic, 2004].

Furthermore, when dragging it can be quite difficult to hit the initial dropping target with the mouse, especially if the target is small (and occluded by the dragged object(s)) [Brewster, 1998]. This problem occurs in particular when the destination target is not highlighted because the dragged object(s), but not the mouse cursor, are positioned over it (Figure 2.3 D) [Gayer, 1989]. Since the object(s) almost fully obscure the destination target, it makes it difficult for the user to see if the target is highlighted, which generally indicates whether the target is going to accept the object(s) or not. Dropping the object(s) in such a situation, does not copy the object(s) into the destination target, but instead positions them directly on top of the target and obscures it even further (Figure 2.3 C).

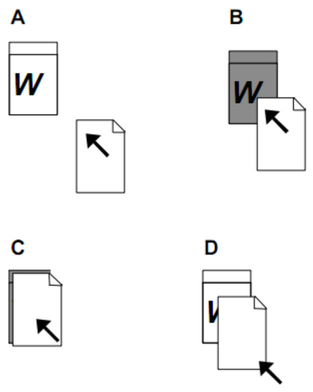


Figure 2.3: Examples of a drag-and-drop interaction [Brewster, 1998]

Dropping errors also occur frequently because the semantics of dropping an object are not always clear and differ from application to application [Dykstra-Erickson and Curbow, 1997]; a drop action might in some instances simply copy the objects, yet in another it might move them.

Another cause of dropping errors is clutching – the act of lifting up the mouse and repositioning it to avoid running out of input area [MacKenzie and Oniszczak, 1998]. Studies have shown dropping errors are more frequent when dragging tasks require clutching because it is harder to maintain the dragging state (holding down the left mouse button) while lifting up and repositioning the mouse [MacKenzie, Sellen, and Buxton, 1991].

Another specific drag-and-drop problem is the so called *drag vs. subselection problem* [Chapuis and Roussel, 2007]. This occurs when users make a text selection that is too large and try to correct it by pressing the mouse button inside the selected text. This initiates a drag-and-drop interaction instead of starting a new selection process [Shneiderman, 2000].

In contrast to drag-and-drop, keyboard shortcuts and menus 'dissociate' the four steps of the copy-paste action sequence, allowing them to be intertwined with other actions like navigation tasks. Nevertheless, and despite the many drag-and-drop problems described above, drag-and-drop 'outperforms' the other options in terms of feedback.

2.3 Underlying Human Factors

Human factors deal with the psychological, social, physical, and biological characteristics of a user in relation to the particular system that is being used. The idea is to use this knowledge of human behaviour, abilities, limitation, and other characteristics to apply it "to the design of tools, machines, systems, tasks, jobs, and environments for productive, safe, comfortable, and effective human use" [Sanders and McCormick, 1993a]. Hence when designing a new interaction technique especially for such a fundamental operation like copy-and-paste, it is pivotal to take the underlying human factors into consideration. Being aware of the human capabilities and limitations leads to better system design.

The underlying human factors influencing the user's performance in a copy-and-paste interaction can be grouped into two main categories: human motor skills (pointing, dragging, and key pressing) and human cognitive information processing (memory, decision making, visual search and interruptions).

2.3.1 Human Motor Skills

A common task in much of human-computer interaction is the need to position a certain entity in space. This may involve (physically) moving a hand or finger to touch an object or (virtually) moving a cursor to a specific object on a computer display using an input device. This task is commonly referred to as pointing [Baber, 1996]. In the case of copy-and-paste pointing plays an important role when selecting objects, defining the destination target and even when invoking copy-paste commands. Characteristics of a pointing task can be closely described (modelled) with the help of Fitts' Law (Section 2.3.1.1). For pointing tasks on a Graphical User Interface (GUI) with an input device (e.g. the mouse) the Control-Display gain plays an important role as described in Section 2.3.1.2.

2.3.1.1 Fitts' Law

Fitts' law [Fitts, 1954] is an effective quantitative method to model user performance in pointing tasks (rapid, aimed movements, where one limb like a hand starts at rest at a specific start position, and moves to rest within a target area) [Mackenzie, 1991]. Fitts' law is the fundamental tool used to design and evaluate interaction techniques and input devices for pointing tasks in GUIs [MacKenzie, 1992; Soukoreff and MacKenzie, 2004]. It describes the movement time taken to acquire, or point to, a visual target on the screen. The movement in Fitts' law is analogous to the transmission of information, which is originally derived from the information theorem [Shannon and Weaver, 1963]. This means for a small, distant target a person needs to transmit more information through the human motor system than for a large, close target. Consequently, it takes longer to acquire the smaller, more distant target. The movement time (MT) (in seconds) to acquire a target of width (W) that lies at distance (or amplitude) (A) is modelled by Shannon's formula (Equation 2.1) [MacKenzie, 1992], where *a* and *b* are two constants determined empirically by linear regression, depending on factors such as input device and population of users [Chapuis, Labrune, and Pietriga, 2009].

$$MT = a + b \underbrace{log_2\left(\frac{A}{W} + 1\right)}_{ID} \tag{2.1}$$

Fitts' law has been shown to hold up for pointing tasks in a variety of conditions, with many different limbs (hands, feet, head-mounted sights, eye gaze), input devices, physical environments (including underwater), and user populations (young, old, special educational needs, and drugged participants). In addition to pointing tasks the Fitts' law also models dragging tasks [MacKenzie et al., 1991]. Dragging, however, has a lower *Index of Performance* than normal pointing, because the increased muscle tension makes dragging slower as well as more error prone.

In conclusion, Fitts' law reveals the somewhat intuitive speed-accuracy tradeoff in aimed human movements: "the faster we move, the less precise our movements are, or vice versa: the more severe the constraints are, the slower we move" [Accot and Zhai, 1997]. Taking this into account, the key factor is to reduce the required travel distance from the starting location to the destination while maintaining a proper target size for clicking.

2.3.1.2 Control-Display Gain

When dragging (e.g. while performing copy-and-paste via drag-and-drop) clutching – the need to lift and reposition the mouse to avoid running out of input area – has been shown to be responsible for dropping errors [MacKenzie et al., 1991]. Clutching is becoming more frequent because the size of displays is increasing rapidly, while the input area remains fixed [Casiez, Vogel, Pan, and Chaillou, 2007]. A way to reduce clutching and therefore dropping errors while dragging is to adjust Control-Display (CD) gain. The CD gain is the mapping between the physical displacement of an indirect mapped pointing device and the corresponding movement of the cursor on the screen [Mackenzie and Riddersma, 1994]. It defines the distance the mouse (or any other input device) has to cover in the physical world to move the cursor on the screen by a specific distance [Blanch, Guiard, and Beaudouin-Lafon, 2004]. The CD gain is a unit free coefficient that can therefore be computed by taking the ratio of the on-screen pointer velocity to the input device velocity (Equation 2.2).

$$CDgain = \frac{V_{pointer}}{V_{device}} \tag{2.2}$$

One of the most popular techniques to modify CD gain is pointer acceleration (PA) [Jellinek and Card, 1990]. PA dynamically adjusts the CD gain using a nonuniform function depending on the input device velocity. Besides PA there are various other dynamic CD gain adaptions, such as expanding targets [McGuffin and Balakrishnan, 2002], bubble cursor [Grossman and Balakrishnan, 2005], semantic pointing [Blanch et al., 2004] and sticky icons [Mandryk and Gutwin, 2008]. All of which are techniques to increase target acquisition performance and reduce clutching by influencing the motor-space through which the input device travels.

2.3.2 Cognitive Information Processing

2.3.2.1 Memory

Imagine you are writing an article. In fact you have written all the individual paragraphs but they still need to be organised and merged together into one complete article. To do this, you copy-and-paste the paragraphs into the right order in a new document. You have just selected a paragraph, copied it to the clipboard and you are about to go back to your target document to paste it at the right place. At this juncture a friend interrupts you and starts talking to you. Afterwards you realise that you have forgotten what you copied to the clipboard only moments ago. The copied snippet has literally been 'bumped out' of your memory. The memory, that is the ability to store information, plays an important role when copy-pasting.

Human memory refers to the processes of the human's brain that are used to acquire, store, retain and later retrieve information. Memory is extremely complicated and above all attempting to investigate it under uncontrolled real-world conditions is frustratingly hard [Baddeley, 1997]. Nonetheless, in recent decades researchers in cognitive neuroscience and cognitive psychology have learnt a lot about memory through controlled experiments conducted in laboratories. It is agreed that memory is not just one unitary system, rather it consists of many systems (Figure 2.4). In fact, according to the Atkinson-Shiffrin model [Atkinson and Shiffrin, 1968] there is substantial evidence that there are at least three very different types of memory storage: sensory memory, working memory and long-term memory.

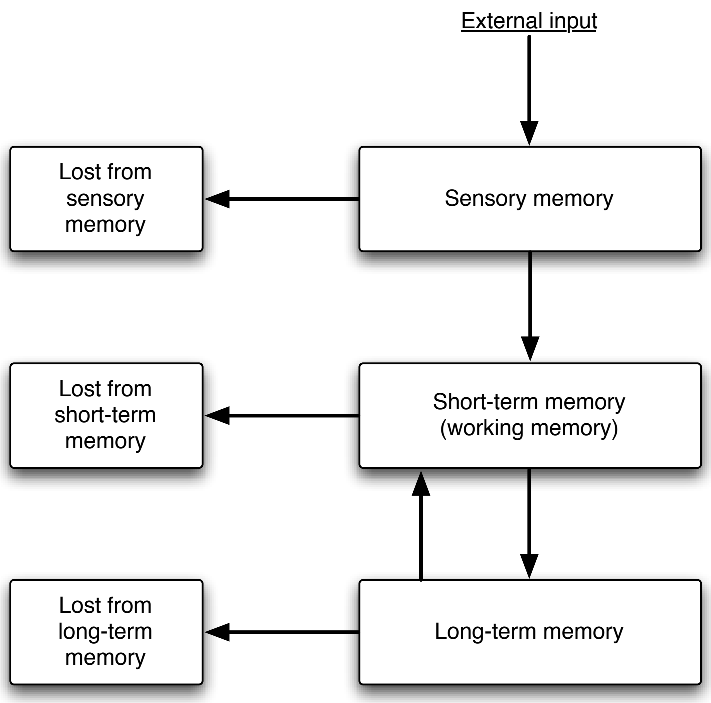

External input
Lost from sensory memory
Sensory memory
Lost from short-term memory
Short-term memory (working memory)
Lost from long-term memory
Long-term memory

Figure 2.4: Atkinson-Shiffrin model adaptation [Atkinson and Shiffrin, 1968].

To apply the computer metaphor to memory, you could say working memory constitutes the general register (CPU's cache) of the cognitive processor (CPU) where all mental operations obtain their operands and leave their outputs [Card, Moran, and Newell, 1980]. Long-term memory, on the contrary, fulfils at most the task of a hard-disk, a large (in theory unlimited) storage unit, where data is stored semantically for later retrieval. Thus, particular limitations of working memory, such as capacity and duration, hold major implications for the design of new systems [Sanders and McCormick, 1993b].

Therefore, out of these three very distinct types of storage, working memory is the one of greatest relevance for the realisation of a new copy-and-paste technique.

Working memory can be understood with the help of the multicomponent model in Figure 2.5, proposed by Baddeley and Hitch [Baddeley and Hitch, 1974; Baddeley, 1986, 1992, 2000]. This model replaces the concept of Atkinson-Shiffrin's unitary working memory system, with specific, active components. It includes a central component, the central executive, and three sub-systems; the phonological loop, the visuospatial sketchpad and the episodic buffer.

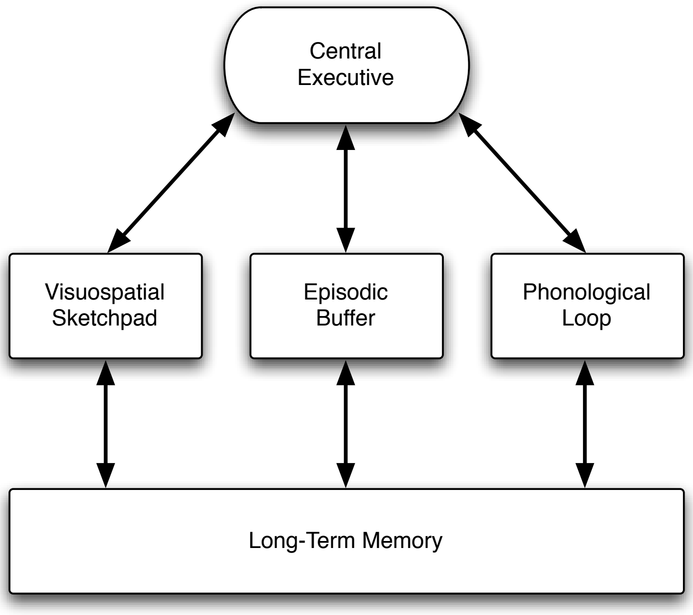


Figure 2.5: Alan Baddeley's Model of Working Memory taken from [Baddeley, 2000].

The central executive component serves as the attentional control system that coordinates cognitive executive processes of the three subsystems. It directs attention to relevant information as well as suppresses attention from irrelevant information. The visuospatial sketchpad holds information in an analog spatial form (e.g. visual imagery), while the phonological loop maintains verbal information in an acoustical form by articulating words or sounds. The episodic buffer links phonological, visual and spatial information in a unitary, episodic representation (in chronological order). Many practical observations can be explained with this model especially with respect to working memory's limitations of capacity and time.

Capacity In 1956, George Miller published one of the most famous articles in cognitive psychology titled "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information" [Miller, 1956]. This article received considerable attention and is known to almost all psychology students. Miller defined the upper limit of capacity of working memory to be around 7 $\pm$ 2 chunks of information. A chunk of information is the unit of working memory space. One chunk can be a single letter, a single numeral or it can consist of several letters and numerals (components) that are strongly associated with one another by their physical as well as cognitive properties. People use this process of chunking information into meaningful entities to expand the capacity of working memory. For example when memorising a telephone number such as 0711446747, the digits can be chunked into the area code (0711), the company's extension (44) and the remaining four digits separately (6 7 4 7). Hence, instead of having to memorise ten chunks, that is each individual digit, this phone number consists of only six chunks (0711-44-6-7-4-7). This means the entire telephone number can be kept in working memory at once. Therefore, chunking reduces the number of items in working memory and subsequently increases the capacity of working memory storage.

Duration Another limitation of working memory closely linked to the capacity is duration – the limit of how long information can remain in working memory. The strength of information in working memory decays over time unless it is periodically rehearsed [Sanders and McCormick, 1993b]. In a maintenance rehearsal process, also known as phonological loop, each (acoustic) item is essentially subvocally articulated. In this way, items will re-enter working memory and will be retained for a further period. The interval of re-activating items thereby depends on the pronunciation time of each item; obviously it takes longer to pronounce a long word than a shorter one. This is also referred to as the word-length effect [Baddeley, Thomson, and Buchanan, 1975]. Thus, the longer words are, the longer it will take to cycle through all the words in maintenance rehearsal, and the more likely it will be that words have decayed. In addition to the word-length effect, similar speech sounds can also make words more difficult to recall. This is called phonological similarity effect [Janet D. Larsen, 2000]. Besides the acoustic properties of words, the semantics (meaning) also have an effect on the capacity of working memory. Previously stored words can interfere with the recall of new words that are similar in meaning [Matlin, 2008].

Researchers have used the serial position effect to examine the recall of items in working memory [Rundus, 1971]. The serial position effect refers to the U-shaped relationship between an item's position in the list and its probability of recall.

When asked to recall a list of items in any order, people tend to begin by recalling the last heard or read items from the list. As displayed in Figure 2.6, the curve shows a better recall for items at the end of the list. This is called the recency effect. These items are still present in working memory and therefore people recall them easiest. However, if there is a delay between 'remembering' and recalling items of the list, the recency effect decreases [Card et al., 1980]. Typically such a curve also shows a strong primacy effect with good recall for items at the beginning of the list. Due to a lower number of items in working memory, more processing time can be devoted to early items. Also, the higher frequency of rehearsing these items effectively causes them to be transferred

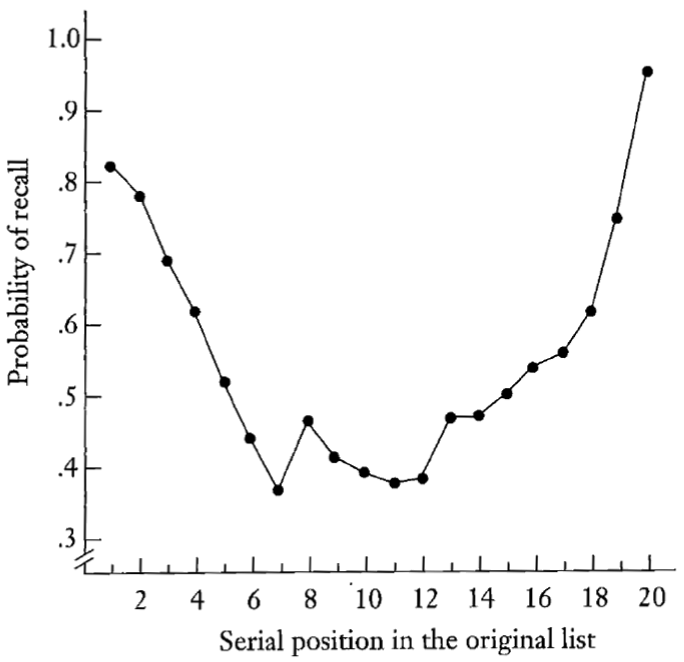


Figure 2.6: The relationship an item's serial position and probability of recall [Matlin, 2008]

to long-term memory. Items in the middle of the list benefit from neither the primacy nor recency effect and thus have the worst recall ratio.

In short, it is crucial to keep working memory load to a minimum. One way to achieve this is to provide cognitive aids in form of visual feedback or visual echoes of relevant information to minimise cognitive overhead during an interaction.

2.3.2.2 Decision Making

Before executing any action with a multiplicity of choices, the user must first choose the action. With regards to copy-and-paste, the user first needs to decide what to copy by selecting one or more objects from a potentially vast number of objects.

Making decisions is time consuming. Making complex decisions is even more time consuming than making simple ones [Raskin, 2000]. The speed with which a person makes a decision is strongly influenced by the number of possible choices (s)he has and the complexity of these choices. The Hick-Hyman Law [Hick, 1952; Hyman, 1953] describes the human decision time *T* required to choose among *C* equally probable choices when optimally prepared (Equation 2.3):

$$T = a + b\ log_2(C) \tag{2.3}$$

The coefficients *a* and *b* are empirically derived constants that strongly depend on conditions, such as how the choices are presented and how familiar the choices are. The decision time increases proportionally (logarithmic) to the number and complexity of choices as Figure 2.7 below shows.

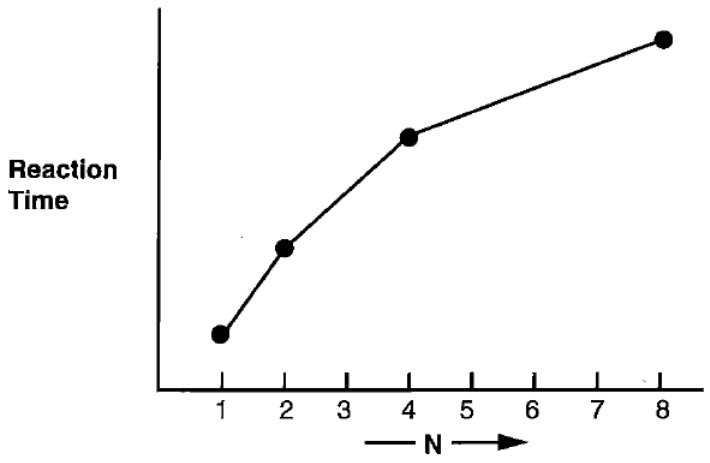


Figure 2.7: Logarithmic increase in Hick-Hyman reaction time as the number of possible choices increases [Sanders and McCormick, 1993b]

Hick-Hyman Law is of logarithmic form because it holds up for cases in which people subdivide the total collection of choices into sections, eliminating about half of the choices each step, similar to 'binary search'. In contrast, when considering each and every choice one-by-one, a linear decision time is required and not Hick-Hyman decision time.

The Hick-Hyman Law can be applied to the selection process of copy-and-paste if the user knows the name of the items (s)he is looking for and the items (whether PDFs, pictures, or other files) are ordered alphabetically. If this is the case sub-dividing strategies can be used, which is the key to Hick-Hyman decision time. If this is not the case, and the user has to visually scan each single item individually (visual search), linear decision time is required.

When modelling decision times, a speed-accuracy tradeoff can occur [Sanders and McCormick, 1993b]. For example, the faster we execute actions, the faster we are forced to make decisions, which is likely to lead to errors. In situations where the consequences of errors are critical, we have to be cautious, and as a result, we will be slower. This is obviously an example of the speed-accuracy tradeoff.

2.3.2.3 Visual Search

Every day from time we get up until the time we go to bed, we spend a great deal of time searching our surrounding environment for information. Whatever tasks we happen to be carrying out require certain information [Peterson, Kramer, Wang, Irwin, and McCarley, 2001]. For example, while driving a car we scan the roadway for other automobiles, traffic signs and pedestrians; in the office, we may look for a specific pen on our desk or for a document we were working on. When performing a copy-and-paste interaction, we may need to visually search for the object we want to copy among a bunch of other objects. A typical example is the need to find a specific picture in a picture library that the user wants to copy to another location. In such a case the user may know the picture's general characteristics or even the picture's content but is not aware of the picture's file name. Hence, the only way of finding the picture is by visually searching the entire library.

Visual search is the process of actively scanning the visual environment for predefined objects or features (targets) among a variable number of competing objects or features (distractors) [Wolfe, Horowitz, and Kenner, 2005]. Search efficiency is an important, often crucial factor. The reaction time taken to respond, whether the target item is present or not, is used as a measurement of search speed. One of the most common factors affecting reaction time relates to the number of visual distractors present in the visual search task. The number of distractors is called the display size. An increase in display size often leads to an increase in reaction time. Therefore plotting the slope of the RT function over the display size gives us a measure for the search efficiency [Kunar, Flusberg, and Wolfe, 2008].

If the target item is identified right away, the search slope, independently of the display size and therefore independent of the number of present distractor items, stays shallow or flat. This is characteristic of feature search [Treisman, 1980]. In feature search, a target item differs from distractors by means of a simple but unique and salient visual feature, like colour, size, orientation, shape or motion. A simple feature may be, for example, a horizontal line among vertical lines (orientation) or a red X among black Xs (colour)(Figure 2.8 on the left). Since the target item differs only in one feature, it has a visual 'pop-out' effect.

In contrast, in a conjunction search task, the target items are defined not only by a single visual feature but by a conjunction of two or more features (e.g., a search for an

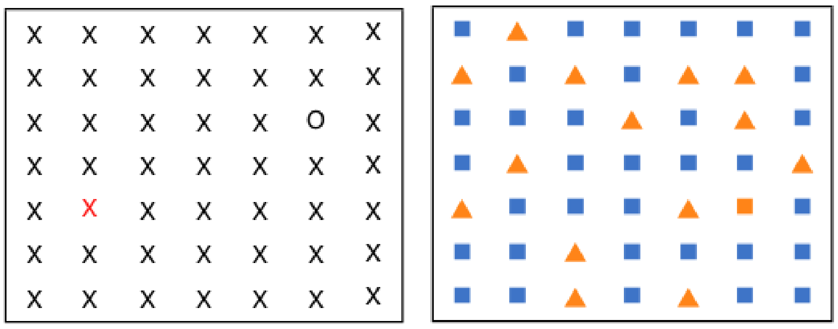


Figure 2.8: Feature search on the left side: The red X as well as the O differ each in one dimension from the distractors and therefore "pop-out". Conjunction search on the right side: The orange square differs in two dimensions (colour and shape) from the distractors.

orange square among blue squares and orange triangles, Figure 2.8 on the right). As a result the target item does not pop out immediately and the reaction time is significantly slower. In fact, the time to complete a conjunction search task increases linearly with the number of distractors added.

2.3.2.4 Interruptions

Consider a scenario where a user wants to cite a part of a text from a webpage in a new article, (s)he is writing. To do this the user selects the text part and executes the copy command, which stores the selected text in the invisible clipboard. Then the phone rings and the user answers the call. After hanging up the phone the user has completely forgotten the initial plan and does not even remember what item is stored in the clipboard – the user's workflow has been interrupted. Situations like this are a common concurrence, especially as modern communication technologies, like phone calls and email notifications, tend to foster frequent task interruptions [Harr and Kaptelinin, 2007]. Moreover, recent studies on workplace multitasking behaviour have shown that people switch tasks or are interrupted every 12 minutes [Mark, Gonzalez, and Harris, 2005].

According to Mark et al. [2005], there are two basic types of interruptions: On the one hand there external interruptions that originate from events in the environment (external sources), such as a phone ringing, a friend entering your room/office, or notifications signalling a received email or instant message (e.g. a pop-up window on the screen). An internal interruption, on the other hand, is when you stop a task yourself on your own volition. Studies show that almost 50% of observed task switches in studies are initiated by the users themselves [Mark et al., 2005; Czerwinski, Horvitz, and Wilhite, 2004].

Interruptions can be beneficial or detrimental depending on the context they occur in. Interruptions that concern the current working sphere are often considered helpful as they might give a positive impulse to the current thinking process [Mark et al., 2005]. Also, people interrupting themselves to take a break in order to 'regain energy' can potentially lead to increases in productivity [Jin and Dabbish, 2009]. In addition, interruptions through email notifications and the like can increase awareness of peripheral information and give direct access to important information.

However, because interruptions divert people's attention away from the task at hand they mostly have a negative impact on the task [Jin and Dabbish, 2009]. This is especially true when interruptions occur outside of one's current working sphere context (not relevant to the current task) or when the task the user was performing at the time involves a high cognitive load, interruptions are considered to be intrusive and disruptive [Mark et al., 2005]. The main problem is the cognitive overhead involved in a context switch between tasks. When a user switches to a different task because of an interruption, (s)he needs to load the context of the new task into working memory. This can be cognitively demanding and time consuming [Cellier and Eyrolle, 1992], but far worse because due to the limited capacity of working memory (Chapter 2.3.2.1), information of the interrupted task gets pushed out of working memory [Y. and D, 1986]. Therefore, when users finally return to the disrupted task, they need to re-load the context of the original task into their working memory. This means they need to recall not only their exact position in the disrupted task, but also all the relevant information to perform the task [Cutrell, Czerwinski, and Horvitz, 2000]. This process of re-orienting to the interrupted task involves a high cognitive cost that requires extra time and can result in a resumption lag [Jin and Dabbish, 2009; Cutrell et al., 2000; Y. and D, 1986].

According to studies, 41% of the time an interrupted task is not resumed right away [O'Conaill and Frohlich, 1995]. However, more than 50% of interrupted tasks are resumed at a later point during the day [Mark et al., 2005]. Tasks that are externally interrupted are more likely resumed and are typically resumed faster than internally interrupted tasks [Mark et al., 2005]. Often people forget at exactly what point in a task they got interrupted. As a result they might resume at a different point from where they had left off. To recall their exact position people rely heavily on cues [Mark et al., 2005]. Cognitive aids, such as 'ToDo-lists', 'Post-it notes' and tying knots in handkerchiefs are common reminding strategies, external cues so to speak, that grab our attention at a time relevant to the task that needs to be performed [Sharp, Rogers, and Preece, 2007]. Therefore, it is essential that systems provide such cues/information systematically about the status of a task. Constant feedback is crucial, especially when users are distracted from the task at hand. The system should be able to inform the users of where they were in a task and what other information is relevant to complete the disrupted task.

2.4 Extensions of Standard Techniques

Copy-and-paste has been studied in many domains, such as programming environments [Kim, Bergman, Lau, and Notkin, 2004; Wallace, Biddle, and Tempero, 2001; Jablonski, 2007] and in graphical [Citrin, Broodsky, and McWhirter, 1994] as well as text editors [Kerr and Stuerzlinger, 2008]. Implications from studies like these have led to numerous attempts to improve copy-and-paste's standard techniques. This section presents these existing extensions and analyses the relative merits of them.

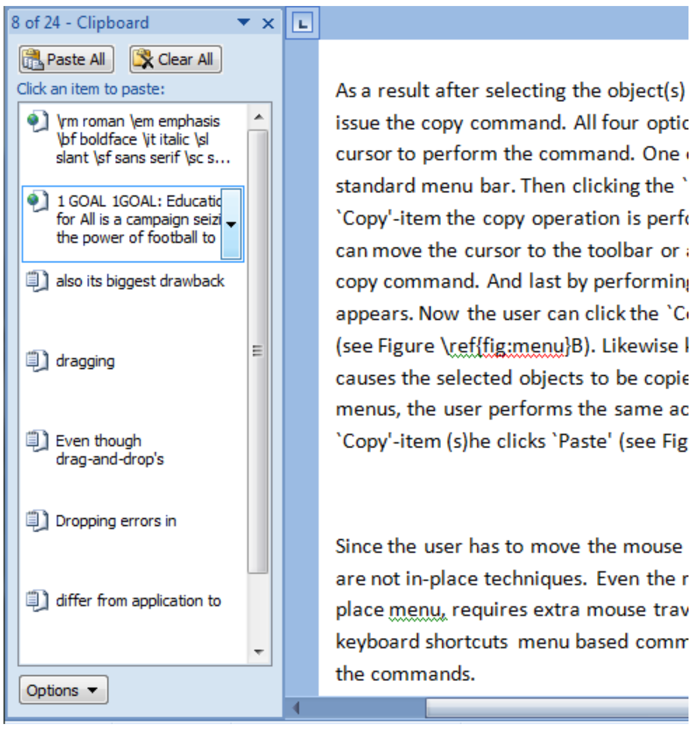
2.4.1 Clipboard Extensions

Keyboard shortcuts and menus are means to access the system clipboard, which has become a standard mechanism across all operating systems to copy-and-paste one single clip between applications. One of the major drawbacks is that, in order to keep it simple, the system clipboard is limited to manage one single item at a time [Stylos et al., 2004]. To counter this limitation numerous third-party clipboard managers have been introduced to extend the standard clipboard's functionality. These extensions serve the sole purpose of allowing the user to manage multiple objects at the same time.

The most widely used clipboard extension is Mircosoft's Office Clipboard[1]. The Office Clipboard works alongside all Microsoft Office applications. As long as an Office program is running, this extension allows the user to copy up to 24 different clips from Office documents or other programs onto the Office Clipboard. When copying multiple clips, the last copied clip is also stored in the system clipboard. The clips on

[1]Microsoft Office Clipboard: http://office.microsoft.com/en-us/word/HA101636021033.aspx

the Office Clipboard can only be pasted into Office documents. To paste one or more clips the user needs to specify the destination in the target document and then select the desired clip(s) from the list of copied clips displayed in the Office Clipboard pane. The Office Clipboard pane, when enabled, is usually displayed on one side of the screen of the running Office application (Figure 2.9). As long as the user stays in that Office application it gives constant visual feedback of the clips copied to the clipboard.



\rm roman \em emphasis \bf boldface \it italic \sl slant \sf sans serif \sc s...

1 GOAL 1GOAL: Educatic for All is a campaign seizi the power of football to

also its biggest drawback

dragging

Even though drag-and-drop's

Dropping errors in

differ from application to

As a result after selecting the object(s)
issue the copy command. All four optic
cursor to perform the command. One
standard menu bar. Then clicking the `
`Copy'-item the copy operation is perf
can move the cursor to the toolbar or
copy command. And last by performin
appears. Now the user can click the `C
(see Figure \ref{fig:menu}B). Likewise
causes the selected objects to be copie
menus, the user performs the same ac
`Copy'-item (s)he clicks `Paste' (see Fig

Since the user has to move the mouse
are not in-place techniques. Even the r
place menu, requires extra mouse trav
keyboard shortcuts menu based comm
the commands.

Figure 2.9: Microsoft's Office Clipboard in action

Having to deal with more than one clip at a time requires more complex interaction techniques than simple two-keystroke keyboard shortcuts like Ctrl-C/V. When using the Office Clipboard, the user can still copy clips to it via keyboard shortcuts or menus. However, to paste a clip other than the last one, the user needs to select the desired clip from the displayed list. To do so, the user needs to move the mouse cursor from the selected insertion point in the document to the Office Clipboard pane and click on the desired clip. This movement of the mouse cursor from the destination to the Office

Clipboard pane constitutes additional motor operations. Furthermore, shifting the focus to the Office Clipboard pane inevitably brings the clipboard extension to the foreground of attention. Such a shift can have a similar effect on a user as an interruption does, as it often disrupts the user's workflow [Faure et al., 2009]. This might be the reason, as a study of daily copy-paste practices suggests, that the Office Clipboard as well as other clipboard managers, such as ClipX[2], Clipdiary[3] or Ditto[4] (that essentially work in a similar way as the Office Clipboard), remain rarely used [Faure et al., 2009].

In contrast to purely software based clipboard managers, Block et al. [2008] have introduced a malleable physical interface that works alongside the mouse and keyboard. This new physical input device gives the user the ability to assign a basic shortcut to each single clip. Hence it preserves the ease of use of a single-item clipboard by maintaining the one-to-one relationship between keyboard shortcut and clip, while allowing the user to manage multiple clips [Block et al., 2008]. As a result, the user needs to have this extra input device (Figure 2.10), which in a physical sense clutters your desk space and in a cognitive sense clutters your brain with the need to memorise which shortcut key is dedicated to which clip. Even if you do memorise each shortcut for each single clip, which will save you from performing certain motor operations, you still need to move your hand to the malleable clipboard to press the respective button.

[2]ClipX: http://bluemars.org/clipx/
[3]Clipdiary: http://clipdiary.com/
[4]Ditto: http://ditto-cp.sourceforge.net/

Figure 2.10: Extending the desktop environment's clipboard with a malleable physical interface to manage multiple clips [Block et al., 2008]

2.4.2 X Window Copy-and-Paste

The X Window system [Scheifler and Gettys, 1986] implements an additional basic copy-and-paste technique, known as X copy-paste, that uses the middle mouse button. In X window systems the user starts an X copy-paste interaction by selecting, for example, a text snippet. Then by clicking on another window with the middle mouse button, that window is not only brought to the foreground, but also the last selected text part is pasted at the insertion point of that window. Essentially this technique minimises the four step action sequence of the copy-and-paste paradigm to only two steps. It works as if the copy command is implicitly executed after each selection and specifying the destination also pastes the selection at the same time [Chapuis and Roussel, 2007]. In addition, under X Window it is possible simultaneously to use the standard clipboard based copy-and-paste functionality accessible via keyboard shortcuts or menus. Obviously X copy-paste minimises motor operations, since the user does not have to press any keyboard shortcuts nor access any menus with the mouse to perform copy/paste commands. Furthermore, in between copying and pasting, it is possible to perform all kinds of auxiliary tasks, even scrolling in another window.

However, on the downside, X copy-paste is not implemented consistently across all applications. For example, some applications may decide to move the insertion point under the mouse cursor before pasting [Chapuis and Roussel, 2007]. Another complication is the inconsistency of input devices. Whereas nowadays most mice have three or more mouse buttons or a clickable scroll wheel that can be used as a middle mouse button, other systems with touch/stylus input or trackpads might not have a physical third button. The middle mouse button then has to be simulated by, for example, pressing a modifier key, which ultimately defeats the purpose of the simple X copy-paste.

X copy-paste's biggest drawback though, is the volatility of the primary selection [Chapuis and Roussel, 2007]. In a scenario, a user selects a URL to paste it into the address field of a web browser. The field of the web browser holds a previous URL, which the user wants to get rid of by highlighting it and pressing the Delete key. Here, pressing the middle mouse button does not paste the originally selected URL, it just 'undoes' the previous delete action and the deleted URL re-appears. Therefore, X copy-paste is only suitable for rather basic copy-paste tasks.

2.4.3 Intelligent Copy-and-Paste

Much of the previous research on copy-and-paste is concerned with making it 'smarter'. **Citrine** [Stylos et al., 2004] is an application that emerged from this research that uses intelligent transformations to extend copy-and-paste. When copy-pasting an address from a webpage/email into a contact manager with conventional copy-and-paste techniques, the user needs to switch windows back and forth for each different data field. Citrine allows the user to paste the structured information in multiple form fields in one single paste operation (Figure 2.11 a-d). To do this it infers the structure of a copied text via numerous parsing techniques. The parsers decide if the copied text contains relevant information. It then adds the copied text in its parsed format to the clipboard and the user can paste it from the clipboard (e.g. into the form of a contact manager) in one single go. To be able to parse and then later on paste information correctly, Citrine often needs to be trained before it works as desired [Bier, Ishak, and Chi, 2006]. Users can teach by example how Citrine should copy and paste certain data [Stylos et al., 2004]. In cases where Citrine parses the information correctly, the user is able to paste information in a single operation and thus mouse movement is reduced significantly. In terms of feedback (as shown in Figure 2.11 b) Citrine displays a little bubble above the taskbar when information is parsed and copied to the clipboard. However, during copying the additional feedback is placed out of focus, so that the user needs to shift focus to see it. There is also no feedback when pasting information.

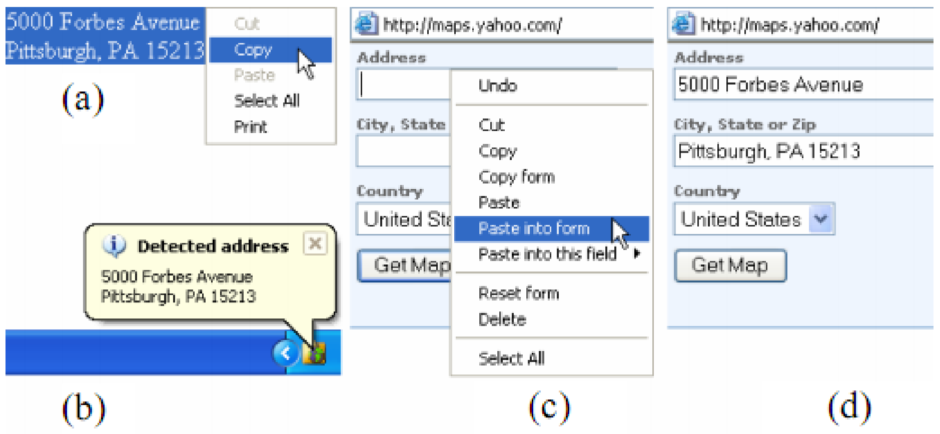
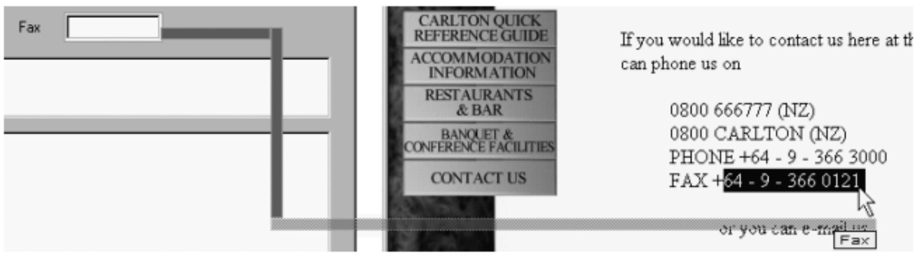


Figure 2.11: Citrine walkthrough [Stylos et al., 2004]

Entity Quick Click [Bier et al., 2006] and the **Stretchable Selection Tool** [Apperley et al., 2000] are two other systems that make use of an intelligent backend. Both extensions reverse the order of the traditional copy-and-paste action sequence. First users have to define the target location and then select the information to be copied. Entity Quick Click [Bier et al., 2006] uses an intelligent mechanism that highlights entities, like single words or phrases, that could be of interest to the user. By clicking on them they get pasted at the previously defined target location. Entity Quick Click only improves copy-and-paste if the algorithm used to extract entities actually highlights the words or phrases that the users want to copy. Hence the whole system is dependent on the intelligence of its algorithm. The Stretchable Selection Tool (SST) [Apperley et al., 2000] uses the metaphor of a 'flexible pipe' to extend copy-and-paste. One end of this pipe is connected to the target location and the other end is dragged with the cursor until connected to the information to be copied. If the user wants to copy, for example, a fax number into a data field of a form, (s)he first clicks into the data field and activates SST. (S)he then moves the cursor, which is connected to the data field via a band and has a text underneath the cursor prompting 'fax' (Figure 2.12), to the desired fax number. By clicking on the fax number, the number gets copied into the data field. The system then automatically jumps to the next data entry field and the text below the cursor changes to the data type of that data field. SST only works in specific scenarios where a form with data entry fields is customised to work with it. Only then can SST jump to the next data field and infer its data type.


Fax
CARLTON QUICK REFERENCE GUIDE
ACCOMMODATION INFORMATION
RESTAURANTS & BAR
BANQUET & CONFERENCE FACILITIES
CONTACT US
Fax

If you would like to contact us here at th
can phone us on

0800 666777 (NZ)
0800 CARLTON (NZ)
PHONE +64 - 9 - 366 3000
FAX +64 - 9 - 366 0121

or you can e-mail us

Figure 2.12: The Stretchable Selection Tool [Apperley et al., 2000]

Another intelligent system is **Drag-and-Guess** [Nishida and Igarashi, 2007], which uses smart predictions to accelerate the dropping task in a drag-and-drop interaction. As the user starts dragging an object, Drag-and-Guess predicts the drop target destination by showing a line connecting the cursor and the predicted target (Figure 2.13). If the target is hidden in a closed folder or beneath another window, Drag-and-Guess makes it temporarily visible. To accept the prediction the user simply needs to release the mouse button and the object flies to the target. If the prediction is incorrect and the user wants to drop the object somewhere else, (s)he can do so by ignoring the prediction and continuing the initial drag-and-drop operation. As soon as the cursor moves away from the predicted target, the line disappears allowing the user a smooth transition to the standard drag-and-drop. Drag-and-Guess saves the user from tedious pointing tasks and complex auxiliary tasks such as making the target visible [Nishida and Igarashi, 2007]. Once again, its effectiveness is completely dependent on the intelligence of the prediction system.

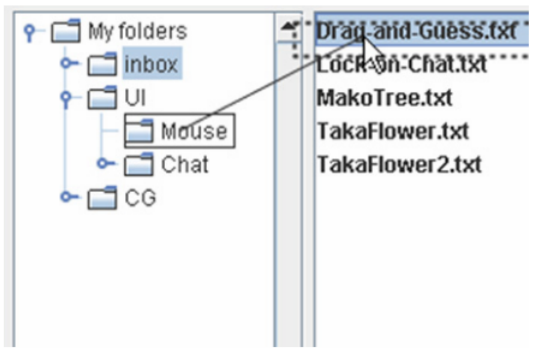


Figure 2.13: Drag-and-Guess in action [Nishida and Igarashi, 2007]

2.4.4 Drag-and-Drop Extensions

The Human-Computer Interaction community has proposed several techniques to improve the usability as well as the functionality of drag-and-drop interactions. Drag-and-drop is a direct manipulation technique. In combination with direct input pointing devices, such as touch/pen input, drag-and-drop interactions require users to physically point on the content they want to interact with. This becomes a problem when content is out of reach, e.g. the target is located too high to reach or on a display that does not support touch/pen input. Also, targets can be located far away from the user, which may even require the user to physically walk over, ending up with a target acquisition time roughly linear to the distance [Guiard, Bourgeois, Mottet, and Beaudouin-Lafon, 2001]. In addition, drag interactions become more error-prone with distance and can be even further complicated by bezels separating screen units [Rekimoto, 1997].

Pick-and-Drop [Rekimoto, 1997] allows users to drag-and-drop data objects in multi-screen and multi-computer environments with pen input. By tapping on an object on the screen with the pen and then lifting the pen off the screen, the user picks up the tapped object and the pen virtually holds the object. Then, the user moves the pen to the desired target destination on the same monitor or another without contacting the display surface. When the pen comes closer to the display surface a shadow of the copied object appears on the screen as a visual feedback. By tapping the screen with the pen the object gets dropped from the pen at the tapped location. According to Rekimoto [1997], this metaphor of Pick-and-Drop is much closer to the real life scenario of picking up an object and dropping it off than the drag-and-drop metaphor of sliding something along a surface. Pick-and-Drop also creates the illusion that the pen is capable of storing objects. However, in reality the storage capability of a pen is 'faked' with a combination of a unique pen-id and a pen-manager server, that manages the transfer of the data-objects over the local network.

Drag-and-Pop [Baudisch, Cutrell, Robbins, and Czerwinski, 2003] provides access to screen content that would otherwise be impossible or hard to reach. As the user starts dragging an object towards a target icon (e.g. a program), tip icons (of compatible type and located in the initial direction of the users drag motion) 'pop-out' at the user's current mouse cursor location. These tip icons are connected to the original icon using a ‘rubber band’, allowing the user to re-identify each tip icon with their original icon (Figure 2.14 (2)). By dropping the dragged object onto one of the temporarily moved tip icons, the respective action (such as deleting the object when dropped on the recycle bin icon) is performed and all tip icons disappear. Therefore, Drag-and-Pop not only allows the user to interact with icons that are hard (e.g. on another display unit) or impossible (e.g. with a pen on an external monitor without touch sensors) to reach, it also minimises overall mouse movement significantly. According to Baudisch et al. [2003], in comparison to traditional drag-and-drop, Drag-and-Pop is superior, particularly on large screens or multiple display unit setups.

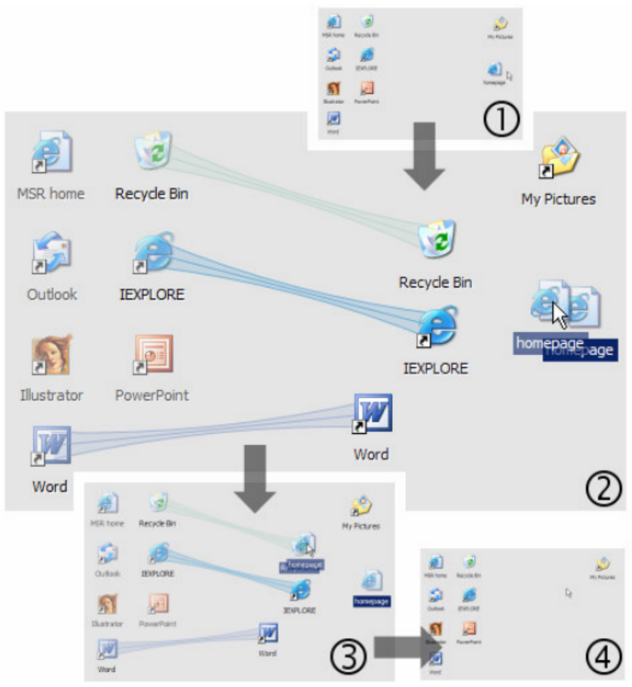


Figure 2.14: Drag-and-Pop: Here the user drops a file located to the right into the recycle bin [Collomb et al., 2005]

Drag-and-Throw [Hascoet, 2003] uses a metaphor of throwing to improve drag-and-drop’s ability to move objects between two displays. Previous throwing models [Geissler, 1998] suffered from bad precision in target acquisition and high error rates. To overcome these limitations Drag-and-Throw provides a real time preview of where the dragged object will come down if thrown. Drag-and-Throw is inspired by shooting an arrow with a bow. With this metaphor in mind, the user first needs to ‘pull’ the to-be-thrown-object in the opposite direction of the destination, aim at the target by looking at the trajectory feedback and then release the mouse button. The trajectory feedback shows the trajectory of the thrown object as a line indicating the throwing path and the exact location on the trajectory where the object would land if the user releases.

Drag-and-Throw’s follow-up, **Push-and-Throw** [Hascoet, 2003], is based on the metaphor of a pantograph [Coxeter, 1989]. In contrast to Drag-and-Throw, Push-and-Throw is initiated by moving an object towards the target rather than away from it. As soon as the user moves an object, the take-off area (a miniature representation of the desktop space around the original object) is shown (Figure 2.15 (2)). Dragging the object inside this take-off area causes a translucent copy of the original object icon, a so called tip icon, to move to the respective location on the real desktop space (Figure 2.15 (3)). The tip icon is connected to the cursor using a rubber band that shows the trajectory along which the object will be thrown when released. Both throwing techniques were found to offer better user performance than traditional drag-and-drop with a slight advantage for Push-and-Throw over Drag-and-Throw [Hascoet, 2003].

Push-and-Pop [Collomb et al., 2005] combines the strength of Push-and-Throw and Drag-and-Pop by removing the need for continuous re-orientation. Push-and-Throw as well as Drag-and-Pop reduce the distance to the target in motor space. Even though the underlying mechanisms are similar, they are visually quite different: Push-and-Throw moves the pointer all the way to the target by showing a rubber-band-connected tip icon in target space. In contrast, Drag-and-Pop moves rubber-band-connected tip icons of potential targets towards the pointer and therefore right into the motor space. Essentially both interactions require the users to re-orient themselves, even though Push-and-Throw

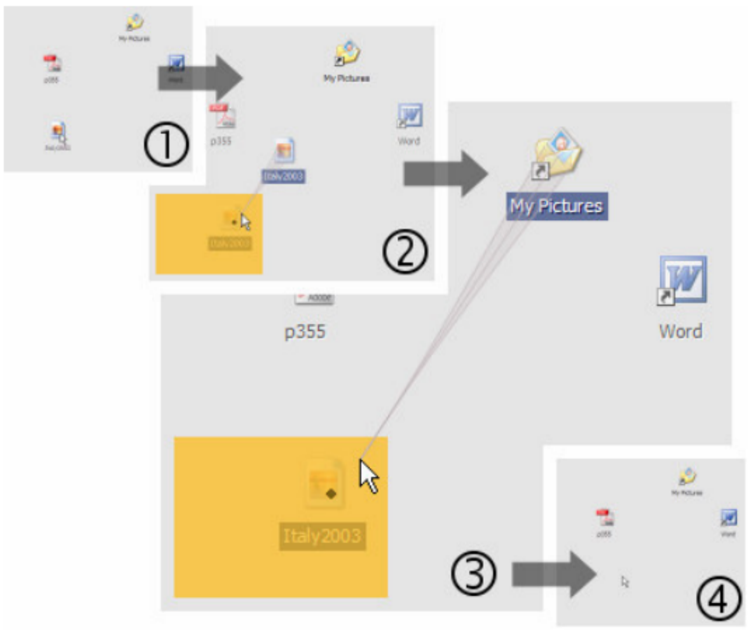


Figure 2.15: Push-and-Throw: The user is dropping the image located in the bottom left into the "My Pictures" folder located at the top right [Collomb et al., 2005]

additionally requires the users to constantly monitor the screen [Collomb et al., 2005]. Push-and-Pop overcomes these limitations by showing the tip icons of potential targets in the take-off area right in the motor space (Figure 2.16 (2) & (3)). Thus no further reorientation for the user is necessary when dropping the object onto the target icon. In Collomb et al. [2005] study, Push-and-Pop outperformed all other techniques, including traditional drag-and-drop as well as all other drag-and-drop extensions described in Section 2.4.4 before.

All of the aforementioned drag-and-drop extensions were designed to improve the efficiency of simple target-pointing tasks with distant targets or on multi-screen environments by minimising motor operations when using touch/pen input [Kobayashi and Igarashi, 2007]. This is essentially done "by temporarily turning the pen/touch input, inherently a direct pointing device, into an indirect pointing device in order to traverse distances faster and to be able to reach locations further away or on different screen

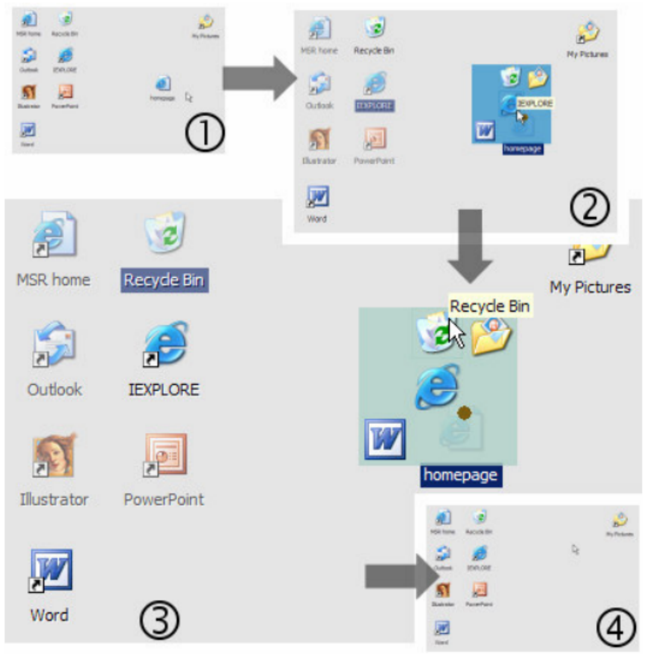


Figure 2.16: Push-and-Pop: The user drags the homepage icon to the tip icon of the recycle bin in the take-off area and drops it. [Collomb et al., 2005]

units" [Baudisch et al., 2003]. However, these techniques do not overcome drag-and-drop's biggest limitation, namely, the inability to perform auxiliary tasks such as window switching while dragging. In contrast, Boomerang [Kobayashi and Igarashi, 2007] and Fold-and-Drop [Dragicevic, 2004] focus on the ability to perform auxiliary tasks while performing drag-and-drop.

Boomerang [Kobayashi and Igarashi, 2007] allows the user to suspend a drag-and-drop operation using a throw-and-catch metaphor. While dragging, the users can throw the dragged object by releasing the mouse button and continuing to move the pointer at the same time. The speed of the mouse movement determines whether the dragged object is thrown or dropped. When thrown, that is, when the speed exceeds a certain threshold, the object flies out of the screen (Figure 2.17 a & b) and the location where it was thrown from is marked by an animated, translucent circle. Once dragging is suspended, other tasks, such as scrolling a window, opening a hidden folder in the file system hierarchy or changing the current window, can be performed (Figure 2.17 c). When the users are finished executing other tasks, they move the mouse pointer to the translucent circle to make the thrown object fly back into the screen. By clicking on the flying object the users catch it and the drag-and-drop operation can be resumed normally. While Boomerang preserves the benefits of traditional drag-and-drop in terms of constant visual feedback, it also allows the user to intertwine the drag-and-drop interaction with other actions like navigation tasks. In addition, it is possible to group drag-and-drop operations by dropping an object on to the translucent circle of another thrown object. All added objects of one group are arranged around the circle at the centre of the group. By dragging this circle, the user can perform a drag-and-drop interaction on all objects of a group at the same time. Therefore, grouping allows the user to copy objects from different folders into one shared folder more efficiently [Kobayashi and Igarashi, 2007]. Since the user does not need to drag each object one by one, it reduces motor operations significantly.

Boomerang and traditional drag-and-drop give constant visual feedback while dragging and dropping an object. However, Boomerang does not give sufficient feedback when a drag-and-drop operation is suspended. When an object is thrown out off the screen, there is only an animated, translucent circle as visual feedback that indicates a drag-and-drop operation was suspended. Just by looking at it, it is not clear which drag-and-drop operation was suspended. The only way of receiving considerable visual feedback is by actually interacting with it. Pointing with the cursor into the circle shows the thrown-off object flying back into the screen. On top of this, if a user chooses to suspend more than one drag-and-drop operation, the screen gets cluttered with translucent circles and it also becomes even more complex to find the circle of the suspended

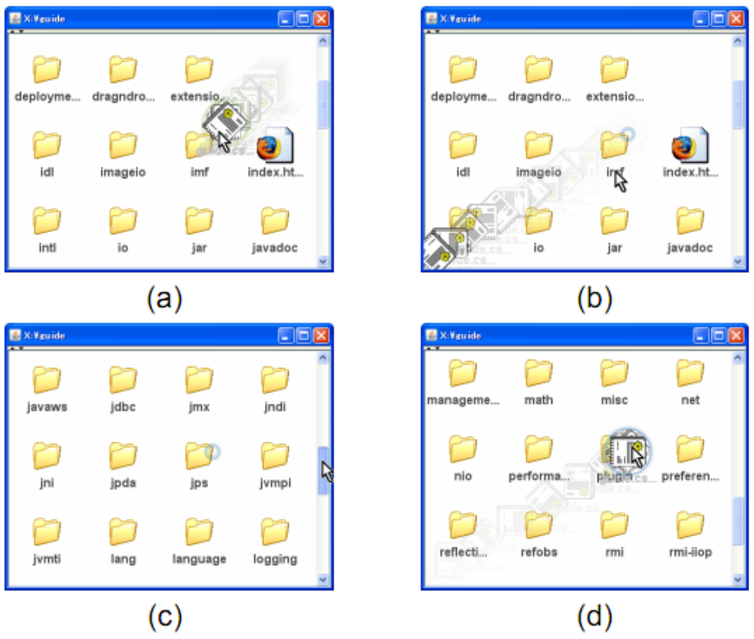


Figure 2.17: Boomerang: (a) Throwing the icon suspends the drag-and-drop interaction and (b) the thrown object flies out of the screen. [Kobayashi and Igarashi, 2007]

operation (s)he wants to resume. Furthermore, it is impossible to cancel a suspended Boomerang drag-and-drop interaction.

Fold-and-Drop [Dragicevic, 2004] uses a paper metaphor and crossing based interactions to fold back windows while dragging an object. Drag-and-drop interactions are done in the usual way, except folding interaction can be performed while the mouse button is pressed to leaf through windows until the target window is found. Each time the mouse pointer leaves a window while dragging an object, a small transient fold appears. This fold appears only briefly before springing back unless the mouse cursor, with the dragged object still attached to it, crosses back into the fold to confirm it. When confirmed (Figure 2.18), the fold can be pushed in order to reveal the window behind it. When a fold is continuously pushed until only a small part of the window remains visible, the whole window disappears. In this manner, multiple windows can be folded successively and at the same time. To unfold a window the user has to go around the fold and push from the inside till the window is completely unfolded. As soon as an object is dropped or the drag-and-drop operation is cancelled (e.g. by a right-click) all windows spring back to their original state. With Fold-and-Drop it is possible to flip through windows while performing a drag-and-drop interaction [Kobayashi and Igarashi, 2007]. Even though it is possible to switch to another window while dragging, it is not possible to manipulate the content of that window, for example, in the form of scrolling to the right location. On top of this, there is quite a lot of additional mouse movement for the folding interactions necessary to leaf through different windows.

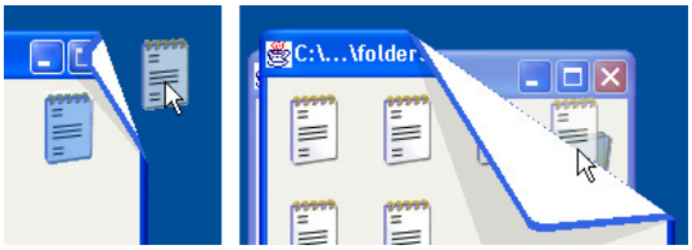


Figure 2.18: Fold-and-Drop: Pushing a fold and dropping the object [Dragicevic, 2004].

Sonically-Enhanced Drag-and-Drop [Brewster, 1998] uses non-speech sounds in form of Earcons [Brewster, Wright, and Edwards, 1993] as additional feedback during a drag-and-drop interaction. When dragging an object onto a target icon, the target icon is usually visually highlighted to indicate that the object can be dropped off. Although, when the icon of the dragged object fully occludes the target icon, there is essentially no visual feedback that can be seen by the user (Figure 2.3 C) [Gayer, 1989]. To overcome the limitations of visual feedback, Brewster [1998] introduced three different Earcons as auditory feedback: The first Earcon starts playing when the object icon is dragged right over a target and stops when it is moved off, the second is played when the object is dropped successfully on the target and the third one is played when the object is dropped but misses the target destination. Using Earcons significantly reduces the time taken to perform drag-and-drop. This is mostly due to the reduction in time spent with the dragged object held over the target icon. The audio highlight is much more effective than a visual feedback because it cannot be obscured. Therefore, auditory feedback in addition to visual feedback can be beneficial.

2.5 Summary

In this chapter, I first introduced the general copy-and-paste paradigm, followed by a discussion of the standard copy-and-paste interaction techniques. In a next step, I described underlying human factors for copy-and-paste techniques and then evaluated existing extensions of standard copy-and-paste techniques against a set of design guidelines inferred from underlying human factors.

This set of design guidelines consists of five categories, all in relation to copy-and-paste's four-step action sequence; select, copy, specify, and paste.

1. To keep the process of selecting and copying objects cognitively as undemanding as possible, visual feedback of the selected/copied objects at the locus of attention is pivotal.

2. It is important for the user to have the ability to perform auxiliary tasks (such as window management) during a copy-and-paste interaction.

3. Minimising motor operations, in particular tedious pointing tasks, improves the efficiency of a technique considerably.

4. Visual feedback when pasting and after interruptions is crucial to minimising cognitive overhead and making a technique more robust against interruptions.

5. Displaying visual feedback at the locus of attention avoids disruption of user's workflow and shift of attention.

Table 2.1 lists all different reviewed techniques. The ✓or x symbol states whether or not this technique supports the respective category. In some cases the ✓might be surrounded by brackets, which indicates that the specific technique only partly fulfils the requirements of the design guidelines of this category. In conclusion, none of the existing techniques passed all five requirements.

	select/copy	*specify*		*paste*	
Technique	**select/copy (visual) feedback**	**ability to perform aux tasks**	**minimise motor operations**	**paste/after interruption feedback**	**no disruption of workflow**
Keyboard Shortcuts	x	✓	x	x	✓
Menus	x	✓	x	x	x
Drag-and-Drop	✓	x	x	x	✓
Office Clipboard	✓	✓	x	✓	x
Malleable Clipboard	x	✓	(✓)	x	x
X Window	x	✓	(✓)	x	✓
Citrine	(✓)	✓	(✓)	x	x
Entity Quick Click	x	x	(✓)	x	x
Stretchable Selection Tool	✓	x	(✓)	x	x
Drag-and-Guess	✓	x	(✓)	x	(✓)
Pick-and-Drop	✓	x	✓	x	✓
Drag-and-Pop	✓	x	✓	x	✓
Drag-and-Throw	✓	x	✓	x	✓
Push-and-Throw	✓	x	✓	x	✓
Push-and-Pop	✓	x	✓	x	✓
Boomerang	✓	✓	✓	x	x
Fold-and-Drop	✓	(✓)	x	x	x
Sonically-Enhanced Drag-and-Drop	✓	x	x	x	✓

Table 2.1: Comparison of all existing techniques

CHAPTER 3

Bag-and-Dump

The *copy-and-paste paradigm* is a fundamental service of today's computer systems that is used by millions of people several times a day. However, existing copy-and-paste techniques have two major limitations:

1. Path length
2. Interruption volatility

This chapter introduces **Bag-and-Dump**, a new copy-and-paste technique, that addresses these limitations.

3.1 Scenario

Chapter 1 presented a brief scenario, which is revisited here for more precision and detail. Anna, an average computer user, wants to compile a slideshow of the best photos she took over the last five years. All of her photos from 2006 to 2010 are organised in a hierarchical structure, ordered by the year they were taken. Her task involves choosing photos from various folders and organising them into one destination folder. For this she requires the *copy-and-paste paradigm*.

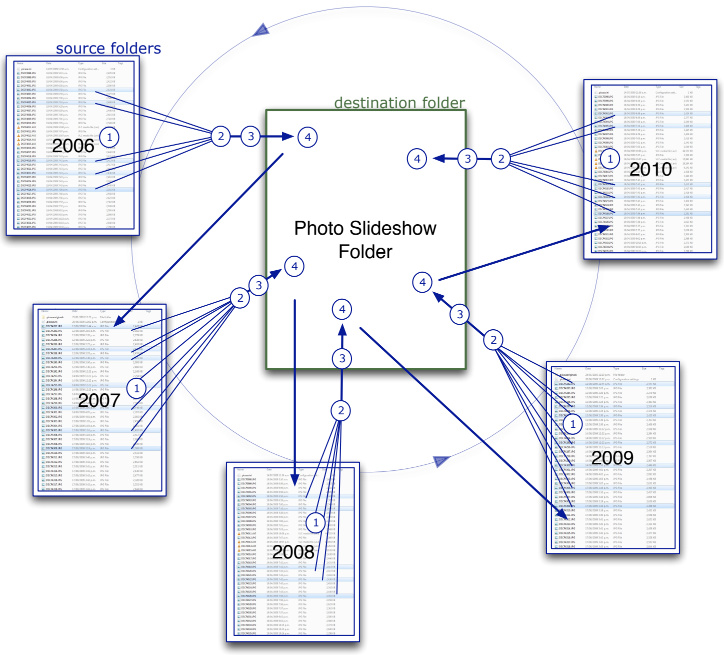


Figure 3.1: Scenario: Creating a photo slideshow out of photos from the last five years using copy-and-paste.

She starts off by creating the destination folder ’Slideshow’. She then goes into the folder of the first year, 2006, (1) selects the best photos of that year by moving the mouse cursor onto the desired photo and then taps the left mouse button while holding down the Ctrl key to add each photo one by one to the selection set. (2) In a next step she invokes the copy command by activating the keyboard shortcut Ctrl-C. After copying the selected photos to the clipboard, (3) she navigates back to the destination folder ’Slideshow’. (4) There she pastes the copied photos by pressing Ctrl-V, which executes the paste command. Then she returns to the source folders and goes into the folder of the next year, 2007. Once again she selects the best photos, copies them, moves to the destination folder and pastes them there. She ends up repeating this action sequence for each year, moving back and forth between source folders and the destination folder. As a result she has to traverse a path of *2n-1* nodes, where *n* is the number of different source folders.

3.2 Design Principles

The previous scenario (Figure 3.1) is one example of a common problem that applies to any domain of copying and pasting across contexts, e.g. with text, images, icons, and so on. Regardless of which copy-and-paste technique is used, substantial mouse movement is involved. Is such a long path of mouse movement really necessary or could it be reduced?

The previous chapter outlined many of the design considerations that need to be taken into account when designing Bag-and-Dump. The five main design principles derived from the table in Chapter 2.5 are:

- Minimise mouse movement
- Ability to perform auxiliary tasks
- Minimise cognitive load by providing sensory feedback
- Support re-orientation after interruption
- Easy to learn (learnability)

All of these principles take into account the limitations of existing techniques. Therefore they will be key points to consider when evaluating Bag-and-Dump's improvement on existing techniques.

3.3 Design Idea

Copy-and-paste is inefficient. Regardless of which existing copy-and-paste technique is being used to copy objects across multiple contexts, it is necessary to go back and forth between different sources and the destination (Figure 3.2 (a)). This means the user has to traverse a path of the length *2n-1*, which constitutes a lot of mouse movement. Imagine it was possible to carry selections across contexts so that the user could go from source to source (Figure 3.2 (b)), select all desired objects, issue the copy command once, go to the destination and paste all copied objects in one go. The outcome would be a path of length *n*, which essentially halves the previous mouse movement path.

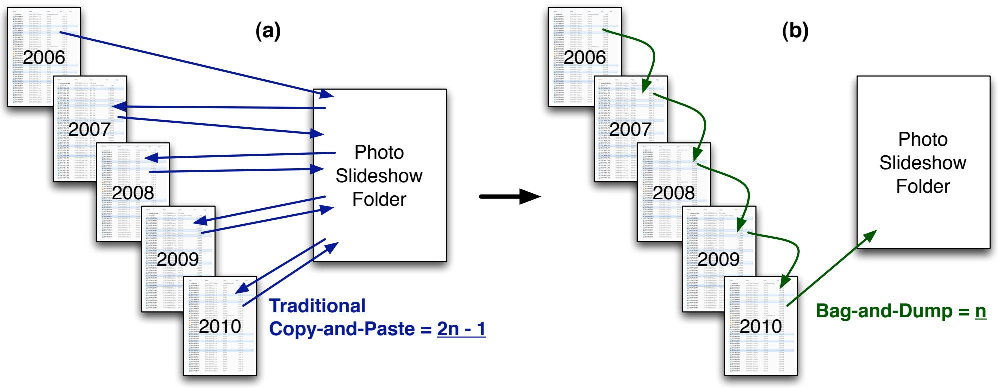


Figure 3.2: Mouse movement of traditional copy-and-paste vs Bag-and-Dump

The idea of **Bag-and-Dump** is to add functionality to copy-and-paste while maintaining its simplicity. The simplicity of the copy-and-paste paradigm lies in its logical four-step action sequence:

1. Select one or more objects,
2. invoke copy command (e.g. by pressing Ctrl-C),
3. specify destination,
4. issue paste command (e.g. by pressing Ctrl-V).

This action sequence follows the noun-verb interaction paradigm, whereas the user first identifies objects and then applies a certain action on them [Raskin, 2000]. Each of these steps can be performed individually, which makes it is possible to intertwine this sequence with other actions and still enable the user to complete the original copy-paste action sequence.

Consistency can facilitate ease-of-use as well as learnability [Jakob Nielsen, 1994]. Therefore, staying consistent with this well-known and simple four-step action sequence can make the transition from traditional standard copy-paste techniques to Bag-and-Dump easier. This underlines the basic idea of relying on an established interaction paradigm and enhance it with new functionality.

Figure 3.3 shows the traditional action sequence of a copy-paste interaction with one slight alteration; the additional green arrow from State S1 to State S3. When using Bag-and-Dump, a user begins by (S1) selecting one or more objects at the current context. In contrast to the common copy-and-paste action sequence and according to the design idea described before, in the next step (S3) the user can move on to another, different context and (S1) select one or multiple objects there as well. This selection process (S1+S3), represented by the additional green arrow in Figure 3.3, can be repeated countless times before (S2) the user finally invokes the copy command and (S3) moves on to the destination (S4) to paste the whole selection.

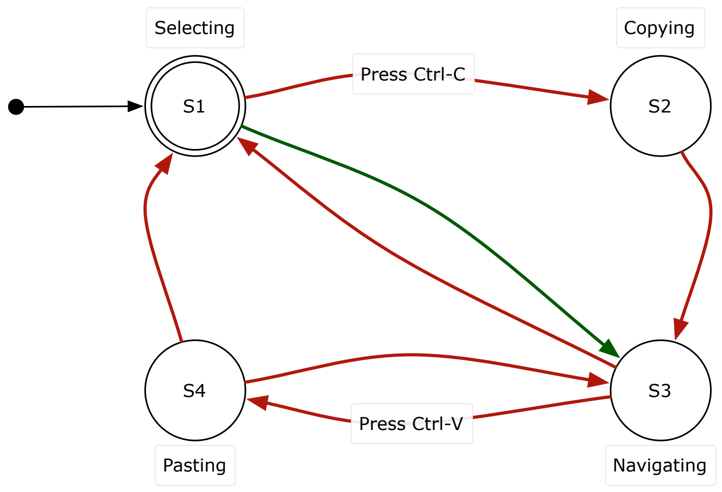


Figure 3.3: A state-diagram representing the action sequence of the copy-paste paradigm with the additional green arrow illustrating the ability of Bag-and-Dump to go from context to context before invoking the copy-command.

Keeping each of these steps separate preserves the ability to perform auxiliary tasks between copying and pasting. This gives the user the freedom to copy an object without knowing the objects pasting destination yet. It is also not necessary to align windows beforehand as it is with drag-and-drop.

3.4 Selection Mode

When using the current copy-and-paste paradigm, the user can select multiple objects by changing selection modes. Changing modes is done by continuously holding down a modifier key while selecting items (as described in Chapter 2.2.1). Pressing Ctrl enables the user to select multiple single items that are then added to the overall selection set. When pressing Shift the user can select a range of items by first selecting the start item of the range and then the last one. These existing selection modes (Ctrl, Shift, Ctrl+Shift), however, only provide the ability to select multiple items at one context. Based on the previously introduced design idea, users should be able to move from context to context while selecting items. These selected items are all added to an overall selection set that then can be further manipulated (e.g. copy-paste).

This section explains the use of modes. It identifies quasimodal interactions as a versatile tool to change between different modes and then introduces the across context selection mode – a new selection mode that allows the quasimodal selection of items from different contexts.

3.4.1 Modes

A mode is a distinct setting of the user interface, in which a certain interaction can produce one of several different responses depending on the system's current state (setting) [Raskin, 2000]. One of the best-known modes is the CapsLock on a standard computer keyboard, which puts the user's typing into a different mode after being pressed. All typed letters are caps. Re-pressing the CapsLock key returns typing to the regular mode.

Mode errors (as originally defined by Norman [1981]) occur when a user misclassifies a situation resulting in actions which are appropriate for the conception of the situation but inappropriate for the true situation. That is especially the case if the state of the system changes without being noticed, or if after awhile the user forgets about the system's state. In the case of CapsLock, if users do not remember pressing the CapsLock key beforehand, they usually end up discovering that they are in CapsLock mode by noticing that letters are all in uppercase on the screen. Other popular examples are texteditors that are famous for their cryptic key combinations, such as vi and emacs [Poller and Garter, 1984], that could trap users in "seemingly inescapable modes" [Hinckley, Guimbretiere, Baudisch, Sarin, Agrawala, and Cutrell, 2006]. Fortunately the consequences of most mode errors, such as the CapsLock case, are only minor inconveniences and can usually be reversed. However, errors in other situations especially in highly complex systems such as airplanes or nuclear power plants can result in cataclysmic outcomes [Sellen, Kurtenbach, and Buxton, 1992].

3.4.2 Quasimodes

A method for combating mode errors is to employ sensory feedback to indicate the current system mode to the user. Visual feedback (the most popular sensory feedback) may help but is often insufficient. For example, in the case of the CapsLock there is usually a small light on the keyboard, which glows when CapsLock mode is engaged. But this kind of visual feedback will fail for touch typists, who do not look at the keyboard. In general many visual cues, even when placed in prominent view, are not guaranteed to work. According to Sellen et al. [1992] visual feedback may be missed even when the user directly looks at it. Therefore it is important to distinguish between what the users' eyes are looking at and what their attention is on. One can easily choose not to direct attention to visual information and hence ignore it.

In contrast, kinesthetic feedback is (physically) more demanding and inescapable by its very nature. Sellen et al. [1992] states "information delivered through the visual channel is simply not as salient as information delivered kinesthetically". Using the CapsLock key is one way of typing letters in uppercase but it can also be achieved by continuously holding down the Shift key while typing. Holding down the Shift key basically requires the user to actively maintain the mode to type uppercase letters. This kind of mode is known as **quasimode** [Raskin, 2000]. Tapping the CapsLock key is regarded as a proper mode because the system and not the user maintains the mode [Hinckley et al., 2006]. The users' active maintenance of muscle tension in a quasimode is the crucial difference to a proper mode. However, this characteristic also makes quasimodes primarily suitable for temporary modes, as users cannot hold down a key indefinitely Hinckley et al. [2006].

Experiments by Sellen et al. [1992] showed that quasimodes not only drastically reduce the potential for mode errors but they also reduce the cognitive burden that is associated with keeping the system state in mind. If the user's attention is on the mode of the system, potentially no mode errors are made, but then also the attention is not on the work the user is trying to get done. In this sense, using kinesthetic feedback keeps the user's locus of attention on the work and not on the modal state of the system.

3.4.3 Across Context Selection Mode

The new across context selection mode integrates seamlessly with the previously described action sequence. By allowing users to carry selections across contexts, it provides the ability to go from context to context while selecting. Pressing a simple new modifier key, such as **Alt**, invokes the new mode and enables the across context selection. To maintain this new mode and carry on selecting from different contexts, user have to continuously hold down Alt. Thus, the new selection mode is based on quasimodal interaction mechanics.

Basing this new mode on the notion of traditional selection modes goes hand in hand with the idea of using established concepts and enhancing them with new functionality to facilitate learnability. Just like the new mode, the Ctrl and Shift selection modes are both quasimodes. Users have to continuously hold down Ctrl and/or Shift to be able to select multiple objects at one context. Keeping in mind that consistency can assist learnability, it makes sense to have the new selection mode along the same lines as the present quasimodal selection modes.

Quasimodes' kinesthetic feedback (holding down a modifier key) produces salient awareness to prevent simple mistakes. With current systems, when selecting at one single context, for example in one single folder, users only need to hold down the Shift and/or Ctrl modifier keys whenever they want to select multiple items. This allows users to perform navigation tasks inside that specific window (folder) such as scrolling down to items at the bottom of a list. It also gives users the freedom after having worked on something else in another window to return to a selection of previously selected items and add further items. When adding new items to a selection set, the user must (be sure to) hold down the modifier key. As soon as the user clicks into empty space or onto an item without holding down the Shift/Ctrl key, all previously selected items are unselected, or in other words, lost. This means one mistake (accidental click or rather the not-holding down of one of the modifier keys) can cause the loss of all previously selected items and therefore waste invested effort. To rectify that mistake the only option is to re-select each single item of the previously selected set. However, according to the principle of commensurate effort [Runciman and Thimbleby, 1986] actions and their opposites (undoing) should involve a similar effort. In this case, a measure to counter the accidental loss of selected items could be a dialogue asking users to confirm their actions. But this is a delicate balance between ease of use and certainty of intent. Since the selection and also un-selection of multiple items is a very frequent action, a constant confirmation would probably entice users to automatically confirm their actions without actually reconsidering the consequences. In this sense using quasimodes is a compromise that counters breaking the principle of commensurate effort as well as ease-of-use by establishing kinesthetic, that is salient, awareness of the current selection mode.

Quasimodes are a simple and efficient way to change modes. Simply pressing a modifier key and continuously holding it down invokes the mode change. The freedom of being able to perform auxiliary tasks while selecting is an important factor in the case of the across context selection mode. The user needs to be able to perform not only navigation tasks in the same window/folder but also more complex auxiliary tasks, like switching between folders or windows while selecting items. If the system was continuously in this new across context mode, it would be hard to distinguish between a simple navigation action, like opening (clicking on) a folder, and actually adding that whole folder to the selection set. This explains the need for a simple and quick mode switching technique. Using gestures or crossing based mode changing techniques [Fitzmaurice, Matejka, Khan, Glueck, and Kurtenbach, 2008; Hinckley et al., 2006] constitutes a lot of additional motor operations if a mode change would have to be performed each time the user would like to add an item to the selection set and then continue to navigate to another location. Using a traditional modifier key such as Alt in a quasimodal way is much more efficient.

All previous research has emphasised that it is crucial to keep mode changing techniques as simple and as efficient as possible so users do not make unnecessary mode errors or get trapped in inescapable modes [Hinckley et al., 2006]. The new across context selection mode introduces new functionality whilst attempting to retain the simplicity of existing standard selection modes.

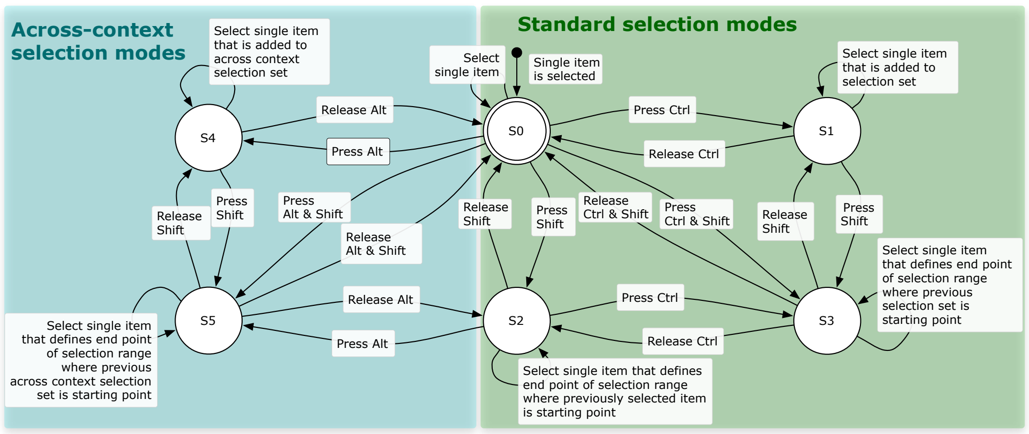

Across-context selection modes
Standard selection modes
Select single item that is added to across context selection set
Select single item
Single item is selected
Select single item that is added to selection set
Release Alt
Press Alt
Press Ctrl
Release Ctrl
S0
S1
S2
S3
S4
S5
Release Shift
Press Shift
Press Alt & Shift
Release Alt & Shift
Release Shift
Press Shift
Release Ctrl & Shift
Press Ctrl & Shift
Release Shift
Press Shift
Select single item that defines end point of selection range where previous selection set is starting point
Release Alt
Press Alt
Press Ctrl
Release Ctrl
Select single item that defines end point of selection range where previous across context selection set is starting point
Select single item that defines end point of selection range where previously selected item is starting point

Figure 3.4: Standard selection modes + across context selection modes state diagram

The state diagram in Figure 3.4 shows that there are two new states. By pressing Alt users reach State S4 and selecting single items from different context adds them to the across context selection set. State S5 can be reached by pressing Alt and Shift, which enables additive range selection at the current context and adds the selected objects to the across context selection set. Essentially States S4 and S5 are quite similar to States S1 and S3. The only difference is, users are able to select items from different contexts and add them to the current selection set.

3.5 Visual Feedback and Interaction

Previous research shows that lack of feedback of the system's state introduces an extra cognitive burden for users [Raskin, 2000]. If there is insufficient feedback, or no feedback at all, users need to constantly keep the system's current state in mind. This means users are required to actively think about two things at once; whatever task they are carrying out and the system's current state. It has been proven though that humans are not efficient when it comes to multitasking, when these multiple tasks require active thought [Medina, 2009]. Studies have shown that humans are only really able to do two tasks concurrently if one of the tasks is automatic such as driving to work while talking on the phone [Rubinstein, Meyer, and Evans, 2001]. In this case driving to work is "automatic" because the driver has done it so often that he has habituated to it. However, as soon as he actively has to think about what he is doing, for example when driving to work on unfamiliar roads during a rainstorm, accomplishing the second task, here talking on the phone, gets harder if not impossible. Therefore current research suggests that humans do not truly multitask, that is simultaneously perform two or more tasks that require active thought. Instead, humans rather cycle through tasks in quick succession [Rubinstein et al., 2001]. Similar to interruptions though, this comes at a price [Rogers and Monsell, 1995]. With each switch, there is a risk of losing one's train of thought. At best, after the switch, the user only requires reorientation to the situation before the switch. Ultimately, rapidly alternating between tasks results in more errors and it takes far longer – often double the time or more – to multitask a set of actions (do them concurrently) than it does to do them sequentially [Wallis, 2006].

So far existing copy-and-paste techniques give some feedback but as discussed in Chapter 2.5, the given feedback is not ideal. Therefore it is pivotal that **Bag-and-Dump** gives constant feedback so that users are always aware of the system's current state at every stage of their workflow.

3.5.1 Bagging (Selecting)

Existing visual feedback while selecting is not optimal and thus increases cognitive load. When selecting multiple different objects using either Ctrl or Shift, there is quasi-modal kinesthetic feedback to remind the user of the current selection mode. There is also visual feedback in the form of colour highlighting of the selected items. These selected items are coloured differently to produce a visual ‘pop-out’ effect, making the selected items distinguishable from the rest. Under Windows, selected items are by default marked in ‘blue’. It seems highlighting selected items should be sufficient visual feedback to minimise the users’ cognitive load. However, when selecting multiple items from a long list of items, that means when navigation tasks, like scrolling, are necessary to select certain items in that list, some of the selected items might not be visible on the screen anymore. So these selected items are no longer visually present, but semantically they are still part of the selection set. Since users cannot see all of the selected items they have to keep track of them ‘in their heads’. The lack of visual feedback on items that are no longer present on the screen introduces an extra cognitive burden for the users.

3.5.1.1 Semantic Cursor

Results have shown that the combination of kinesthetic and visual feedback significantly improves performance and is effective in preventing mode errors [Sellen et al., 1992]. Graphical editing applications like Paint and Photoshop use semantic cursors, e.g. lasso, magnifier glasses, text and so on, to communicate the current tool being used. As the name “semantic cursor" implies, the cursor icon itself is used to communicate a certain semantic meaning. Since the cursor is in the user’s focus most of the time, it is the perfect means to give constant visual feedback about the system’s current mode. That is why Bag-and-Dump utilises the idea of a semantic cursor to visually communicate the current selection mode. In addition, Bag-and-Dump makes use of kinesthetic feedback in the form of quasimodes to remind users of the current mode.

As soon as the user switches into the new across context selection mode, the semantic cursor for the new selection mode is displayed. This semantic cursor looks like a ‘bag’ that is attached to the mouse cursor. This ‘bag’ visually represents the selection set, which the user fills by going from context to context and selecting new objects. Metaphorically speaking it is like a person going from shop to shop buying new things and putting them into a shopping bag (Figure 3.5). While doing so the shopping bag naturally becomes heavier and bulkier. Likewise the ‘bag’ not only serves as visual feedback in terms of displaying the system’s current mode, it also gives constant visual feedback regarding the actual content of the selection set.

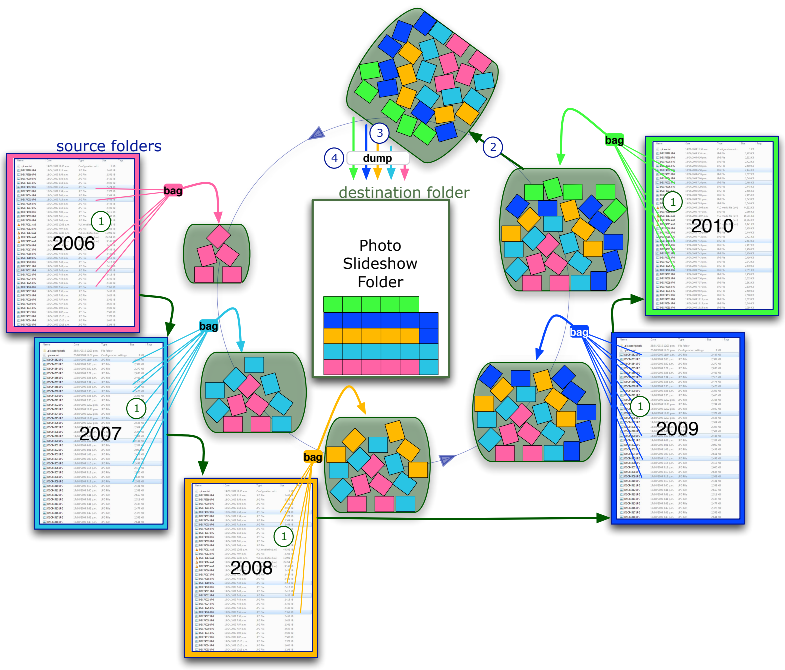


Figure 3.5: Bag-and-Dump is based on the “shopping bag metaphor”

3.5.1.2 Bag's Visual Appearance

There are a few factors that need to be taken into consideration for the visual appearance of the 'bag':

- simple look-and-feel
- clear communication of the content
- readability
- visually/aesthetically pleasing (subjective)

First of all it is important that the bag clearly communicates its purpose and content. It should have the look-and-feel (the so called affordance) of a bag that is attached to the mouse cursor so that just by looking at it, its functionality is apparent to the user [Norman, 2002]. Thus in terms of visual appearance the 'bag' like semantic cursor looks like a bag that is 'dangling' from the cursor (similar to a speech bubble used in comics). The bag is of rectangular shape with roundish corners and a long pointy triangle in the top left corner that connects the bag with the mouse cursor (Figure 3.6). Similar to the metaphor the bag's size depends on its content. The bag is always displayed when the across context selection mode is activated; if nothing has been selected yet an empty bag of minimum size (150pixel width and 50 pixel height) is displayed. The bag is translucent, so that the information in the background is still visible, providing the users the necessary overview of the context while selecting. At the same time the bag's translucency must not come at the expense of readability. Therefore the bag's translucent background is sufficiently rich in contrast so that the displayed items are easy to read.

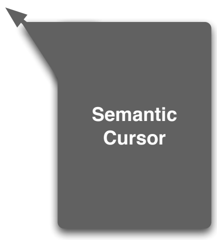


Figure 3.6: Bag-like Semantic Cursor

As soon as objects are added to the selection set they are visually displayed in the 'bag'. There are many ways the content of the 'bag' could be visualised. Bag-and-Dump uses a combination of a list and vertical timeline/stack visualisation for the bag's content. In a list, which is a common method of visualising different objects, each row represents one object. Likewise the items in the bag are each presented in their own row. The item's name and its file type-icon (or in case of a picture its thumbnail) are the primary means of identification for an item. Therefore each row displays the item's type-icon followed by the item's name as shown in Figure 3.7. When the item is not a file but, for example, a text snippet, then the first 30 characters of the copied text are displayed instead of a filename. Items are displayed in the order they were added to the selection set. This is like putting items onto a stack which essentially resembles a vertical timeline as shown on the left in Figure 3.7.

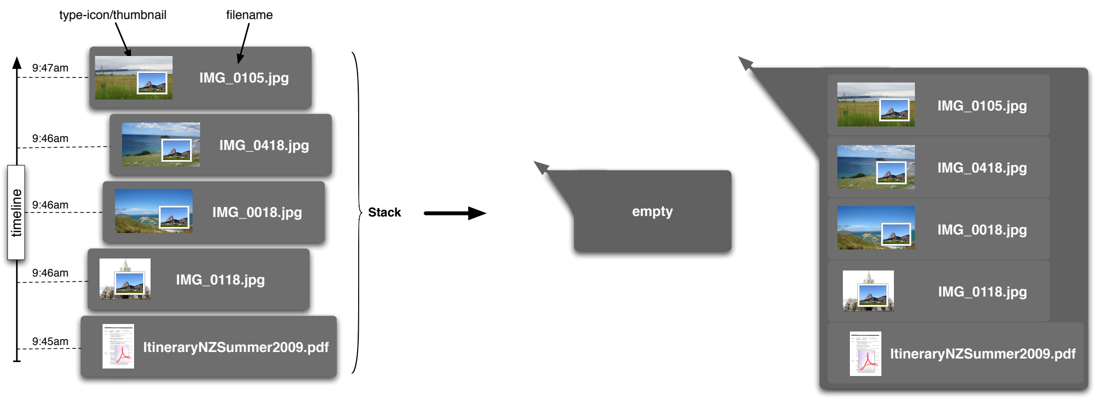


Figure 3.7: Visualisation of Semantic Cursor's content

In general, items are presented as flat, individual objects. This means, for example, an item does not exhibit any reference to its location in the hierarchical filestructure or to any other meta-information. This keeps the idea of an item as simple as possible.

When the bag contains many items and each item is presented on one row, the size of the bag increases steadily with the number of items. A big bag would not only occlude a lot of the screen estate, but also if the bag is bigger than the screen itself some items of the bag would not be visible anymore because they are outside of the screen's borders. Therefore, it is necessary to adapt the stack-list visualisation to the available screen space. To guarantee that the user does not lose context of the current selection workflow, it is necessary that the last added items are always fully displayed with file type-icon/thumbnail and name. Therefore only the presentation of items lower in the stack (items added first to the selection set) can be altered.

There are numerous ways of shrinking the size of the visualisation of these items. Scrolling and zooming could be obvious solutions for it. However, the necessity to directly interact with the bag would inevitably bring (the interaction with) the bag to the foreground of attention. This shift of attention could disrupt the user's workflow (similar to the Office Clipboard in Chapter 2.4.1) and could have an equally detrimental effect as interruptions. To prevent this, it is important that the visualisation displays the bag's entire content without the necessity for the user to directly interact with it.

Two possible ways of minimising the visualisation's screen space usage of the remaining items while always fully displaying the last added items could be:

1. Grouping: Remaining items are grouped either by e.g. their file type or their location in the filestructure.
2. Grid: Remaining items are displayed in a grid, whereas each item is only represented by its file type-icon/thumbnail.

The first idea partly contradicts the idea of displaying items as flat, individual objects, since items are grouped according to some of their meta-information such as location or filetype. However, the main purpose of the items in the lower part of the stack is to provide a visual cue helping to overcome loss of context. Grouping these items would serve this purpose and the main visualised items in terms of the upper part of the stack would still be displayed as flat, individual objects. These two options should be further investigated in future evaluations.

The visual presentation of the bag's content could also be enhanced by giving other feedback, such as haptic feedback. Pressing the Alt key already gives quasimodal active kinesthetic feedback. Imagine if, in addition, as the bag gets fuller, the mouse cursor's movement speed slows down, giving the impression that the bag is getting heavier and heavier. This type of haptic feedback could be regarded as passive kinesthetic feedback; users receive haptic feedback but do not need to perform any physical action themselves. This kind of feedback could add another dimension of sensory feedback to the interface.

3.5.1.3 Bagging Interaction

In terms of the interaction, the bag is brought up when pressing Alt to activate the across context selection mode. After pressing Alt the first time, the bag is continuously displayed. However, as soon as users click into empty space or on any item without holding down Alt the bag disappears and its content is dismissed. When pressing Alt and selecting an item at the same time, then that item is 'bagged' and thus displayed in the bag. When decided that all desired objects are selected, the user can copy the selected items to the clipboard by pressing Ctrl-C. The idea here is that the user only needs to learn how to use the new selection mode and then can combine it with familiar standard copy command interaction techniques to complete the copy-paste interaction. As soon as the user presses Ctrl-C, the bag's content is copied to the clipboard and the bag disappears.

3.5.2 Dumping (Pasting)

Consider a similar scenario to the one before. Here, Anna selects a few photos and executes a copy command, which stores the selected photos in the invisible clipboard. Then she gets interrupted, for example, by a phone call. Such an interruption could cause her to lose her train of thought, in particular, when the task the user was performing at the time involves a high cognitive load [Harr and Kaptelinin, 2007]. In this case when Anna got interrupted, she was navigating to the destination folder for the copied photos. After the phone call she returns to her copy-paste task but she cannot fully remember what she was doing before. She does remember she was copying photos across but cannot recall what photos are currently stored in the clipboard. Since there is no visualisation of the content of the clipboard and she cannot remember the clipboard's content, the only way to recapture the stored data is by pasting it somewhere. If she pastes the photos into the wrong folder though, she has to re-select the pasted photos and move them to the right location or possibly even delete them. This means additional operations have to be performed. All too often instead of pasting the clipboard's content to find out what is in it, users start all over again. In this scenario, Anna ends up going back to the source folder and again selects a bunch of photos, that are completely different from the ones she copied before the interruption, and then by issuing the copy command she inadvertently overwrites the invisible content of the clipboard [Apperley et al., 2000].

These extra actions to either recover the clipboard's content or re-issue the copy-command are necessary because there is no visualisation of the clipboard's content. Essentially the non-existence of any cues to the content of the clipboard means the users have to think about the content of the clipboard as well as think about the task they are about to perform. This introduces an additional cognitive burden that could be prevented through sufficient feedback.

Therefore, not only is constant visual feedback crucial when selecting, but visualising the clipboard's content when pasting is also of extreme importance. It is pivotal, though, that the visualisation and the interaction that brings it up does not break the user's workflow. Existing clipboard managers, which visualise the clipboard's content, tend to cause a switch of attention and often even require significant extra mouse travel similar to Microsoft's Office Clipboard (Chapter 2.4.1) [Faure et al., 2009]. To avoid these problems, the content of the clipboard needs to be displayed at the user's locus of attention. Just before invoking the pasting command the user specifies the insertion point by navigating to the desired location with the mouse cursor. As a result the mouse cursor is in the user's focus when pasting objects, so it makes sense to visualise the clipboard's content at the mouse cursor's position. Furthermore, since the mouse cursor is used to determine the insertion point for the pasting operation, it effectively carries the clipboard's content on its back like a shopper carries a shopping bag just before dumping its content somewhere. Thus displaying the clipboard's content as the 'bag' described before seems suitable.

In addition to visualising the clipboard's content, the interaction that brings up the clipboard's content is also crucial. Such an interaction needs to be "carefully designed if one does not want to break the user's workflow" [Faure et al., 2009]. With existing copy-paste techniques, invoking the paste-command means that the clipboard's content gets pasted at the designated insertion point. However, as explained before in some situations it would be advantageous if one could see the content of the clipboard before actually pasting it. Faure et al. [2009] introduced a copy history menu, for which they used timeout delimiters [Hinckley, Baudisch, Ramos, and Guimbretiere, 2005] to smoothly integrate new interaction techniques with old ones. In their case a long Ctrl-V (that means pressing the keys for more than 250ms) brought up the copy history menu. With Bag-and-Dump a long Ctrl-V (>250ms) brings up the 'bag'. Pressing Ctrl-V a second time pastes the clipboard's content. The pasting operation can be cancelled by clicking with the mouse at any point on the screen or pressing any other key on the keyboard. A normal short Ctrl-V would immediately paste the clipboard's content without bringing up the visualisation. As a result the user can decide whether or not to see the visualisation of the clipboard's content before pasting. Providing this kind of freedom is important because if the user executed the copy-command just before invoking the paste command it is quite likely that showing the visualisation would be unnecessary since the user is still aware of the clipboard's content.

3.6 Summary

In summary, Bag-and-Dump's interaction technique attempts to adhere to earlier established design principles. Its across context selection mode allows users to carry selection across multiple contexts and therefore cuts the path of mouse movement compared to existing copy-paste techniques in half. At the same time, users are able to perform auxiliary tasks, which gives them the freedom to perform their tasks the way they want to. Furthermore, Bag-and-Dump introduces additional visual feedback that constantly informs the user of the system's current state. This additional visual feedback (when selecting, as well as when pasting) not only minimises the users' cognitive load but it also supports users in re-orientating after interruptions. One of the main characteristics of Bag-and-Dump though is the re-use of familiar concepts to make it easy to learn. It adopts the concepts of standard selection modes and keyboard shortcuts Ctrl-C/Ctrl-V, two fundamental interaction concepts of the copy-and-paste paradigm, and combines them with additional functionality. Since users are mostly familiar with the interactions of these two concepts they only need to learn and retain a small amount of new information to be able to use Bag-and-Dump. Therefore, by staying consistent with existing concepts and by adding additional functionality, especially in the form of constant visual feedback, Bag-and-Dump aims to facilitate learnability.

CHAPTER 4

Experiment

I conducted an experiment to answer five key questions about Bag-and-Dump's (BND) performance in comparison with two other copy-paste techniques, Keyboard Shortcuts (KEY) and Drag-and-Drop (DND).

Q1. Is BND faster (in terms of trial completion time) than KEY and DND?

Q2. What effect do interruptions have on BND's performance in comparison to the other techniques?

Q3. Are fewer errors made with BND than with KEY and DND?

Q4. Does BND reduce mouse travel distances compared to KEY and DND?

Q5. What is the subjective response to BND?

All three techniques were tested under three different item modes and with and without interruptions. The three item modes are: six items at one context (6@1), six items at three contexts (6@3) (that is two items at each of the three contexts) and six items at six contexts (6@6) (hence one item at each of the six different contexts). Interruptions, in form of mathematical problems that needed to be solved, disrupted the workflow of the participants in half of the tasks.

A good way of comparing different techniques is by using common tasks occurring in everyday computer usage and thereby simulate real scenarios. One frequently occurring task (as described in the scenario in Chapter 3.1), is to copy multiple files from different source folders (in this sense different contexts) into one shared destination folder. This file copy-paste task is the basis of the experiment. In the process of the experiment, the task is repeated numerous times, which requires participants to perform a large amount of copy-paste operations. This is a simple way of obtaining as many copy-paste completion times as possible in a minimum amount of time. Unfortunately the reality of carrying out an experiment of this nature is the synthetic task saturation, which is less likely to occur during normal use.

Performing a file copy-paste task usually requires the use of a file manager application. Most file manager applications, such as the Windows Explorer, use a split window with a treeview of the tree topology on the left side, and a list/tile view showing the content (sub-items) of the selected treeview-item in more detail on the right side. The treeview presents the hierarchical correlations of all folders, that is items with sub-items, in a nested list. The topology of the hierarchy is visualised by using indentation and lines connecting the different items. The Windows Explorer is a popular example of a graphical interface combining a treeview and a listview to navigate/browse through a tree-hierarchy. Hence this widget is called treebrowser.

This experiment used such a treebrowser widget with treeview on the left and listview on the right, to display target, as well as distractor, items. Target items were the items that were supposed to be selected and copied from the treebrowser and then pasted into the destination textfield. Using this analogy of a treebrowser confronted the participants with a familiar graphical interface. This should make tasks using this interface easier to learn since participants did not need learn to use a completely new and unfamiliar interface. Furthermore, in a treebrowser, items were easily able to be displayed in categories and sub-categories. This was a good way of simulating different contexts just like it is with folders and files in a hierarchical filesystem.

As it is in the Windows Explorer, only folders (items that have sub-items) were displayed in the treeview on the left side of the experiment's treebrowser widget. In the treeview, clicking on a mini 'Plus/Minus'-Button located to the left of items with sub-items expanded the item by revealing its sub-items and respectively collapsed the item by hiding its sub-items. Double-clicking on an item with sub-items had the same effect. A single click on an item in the treeview selected the item and also showed its sub-items, if any existed, in the listview on the right. Selected items were highlighted in the Windows Explorer's standard selection colour 'blue'. Whenever an item was highlighted in the treeview, its sub-items were shown in the listview. Therefore when expanding/collapsing or selecting an item in the treeview, the listview of sub-items on the right was updated right away. To select items in the listview on the right all standard selection modes (eg. single selection mode, single multiple selection mode, range selection mode, and additive range selection mode) could be used. Clicking on an already selected item in the listview, de-selected that item. Clicking into empty space in the listview deselected the selected item(s), unless modifier keys, such as Ctrl or Shift were pressed. A doubleclick on an item with sub-items in the listview displayed its sub-items in the listview, and also highlighted the clicked-on item in the treeview.

The goal of this experiment was to compare Bag-and-Dump to the standard copy-paste techniques: Keyboard Shortcuts, Menus and Drag-and-Drop. One of the standard techniques, Menus, was not included in the experiment because Keyboard Shortcuts and Menus are almost the same. They are both based on the exact same four-step action sequence. Their only difference lies in the low level mechanics used to execute the copy/paste commands. Studies [Chapuis and Roussel, 2007] have shown, though, that the use of Menus to execute these copy-paste commands is significantly slower than Keyboard Shortcuts. The difference in terms of the execution time could be modelled with the help of KLM [Card et al., 1980]. Drag-and-Drop, on the other hand, follows different interaction mechanics to Keyboard Shortcuts. Therefore, Keyboard Shortcuts and Drag-and-Drop were both included in the experiment and compared against Bag-and-Dump.

All three techniques were evaluated under three different item modes: 6@1, 6@3 and 6@6. The item modes defined whether all six target items were at one, three or six contexts, whereby a context represented a folder in the treeview. By spreading the items over different contexts, participants were required to go to each context and select the required items there. First of all, this was a way to see if the number of contexts involved in a task influenced the completion times. But even more so, it was a way to compare how these different techniques coped with an increasing number of contexts. At the same time, it was possible to see whether shorter mouse distances (as expected with BND) resulted in faster completion times.

In addition to the item mode condition, techniques were also tested against interruptions. Comparing completion times of interrupted and not-interrupted copy-paste trials was a way of testing the robustness of different techniques against interruptions while selecting items. In the experiment, the interruption had to be significant enough to bump the relevant information required to perform the copy-paste task out of the participant's working memory. Hence, the participants were given a sufficiently difficult mathematical problem they had to solve. When they returned to the disrupted copy-paste task, they not only had to recall their exact position in the task, they also had to recall all relevant information, such as already copied and pasted target items. This process of re-orienting to the interrupted copy-paste task involves a high cognitive cost that usually requires extra time.

4.1 Apparatus

The experimental system was implemented with Python 2.6.5 and wxPython. The experiment ran on 2.6GHz PC running Microsoft Windows 7 with a powerful graphics card connected to a 1680 x 1050, 21" LCD display. The mouse used was a standard optical mouse with two buttons and a clickable scroll-wheel that could be used as a third (middle) button. The default Microsoft Windows 7 mouse acceleration was used.

4.2 Participants

Twenty-two participants (4 female, 18 male) ranging in age from 22 to 44 years (Mean age: 28.7) participated in the study. They were recruited from the Computer Science

Department at the University of Canterbury. The participants can be considered as "expert users", since they were all post-graduate computer science students that reported using computers in average 51.9 hours per week (SD 16.6). Among them, seventeen use Microsoft Windows, three use Linux and two use Apple Mac OS X. Participants received a $10 Shopping voucher as compensation.

4.3 Experimental Design

A repeated measures 3 $\times$ 3 $\times$ 2 factorial design was used. The three factors were the copy-paste *techniques* (KEY, DND, BND), the *item modes* (6@1, 6@3, 6@6) and *interruption conditions* (no interruption/interruption). The main dependent measure was the completion time to perform one experimental trial copying six target items from the hierarchical tree structure of the treebrowser on the left and paste them into the textfield on the right (Figure 4.1). Other main measures were the error rate as well as the distance the mouse travelled during each trial. To further charaterise interactions, the following measures were also analysed: the time it took from starting a trial until selecting the first item, the different selection modes used, the number of left- and right-mouseclicks, the time it took from the point the last item was selected until the copy command was executed, and the number of copy as well as paste operations that were executed.

The experiment consisted of three trial groups, each group consisting itself of a series of 18 trials. Within each group the copy-paste technique was fixed. Thus copy-paste techniques were not intermixed. Orders of techniques were pseudo-randomly counter-balanced across participants following a Latin-square design.

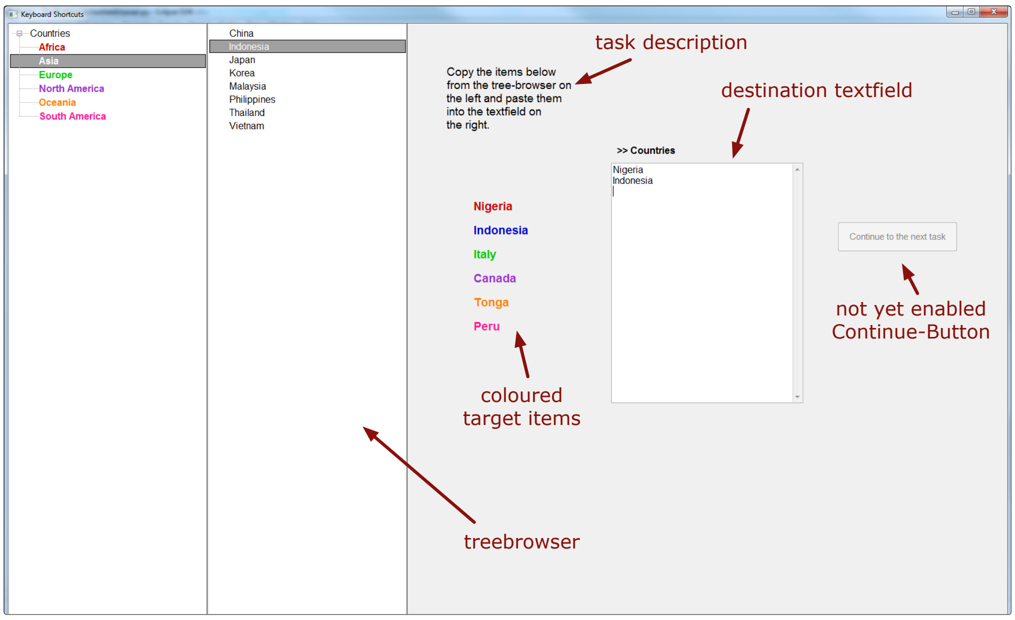

Copy the items below from the tree-browser on the left and paste them into the textfield on the right.

Figure 4.1: Screenshot of the experiment's GUI.

4.4 Method and Procedure

A trial consisted of selecting six items from one (6@1), three (6@3) or six contexts (6@6) in the treebrowser on the left and copy-pasting them into the destination textfield on the right. To start the first trial the participant had to press the Start-Button in the middle of the screen. At any stage of a trial a task description, stating the target items to be copied, was displayed in the middle of the screen. On the left was the treebrowser widget and on the right the destination textfield as well as the yet to be enabled Continue-Button (Figure 4.1). The target items' parent nodes were colour-coded in the treeview of the treebrowser. That means blue coloured target items of the task description were to be found under the blue (parent) item in the treebrowser and so on. Colour-coding the target items was done to prevent the necessity to visually or even semantically search for the target items at each single context. Searching for the target item at each single context in the hierarchical structure could create noise in the experiment's data since participants might have different knowledge of the different categories and would therefore take a different amount of time to find the target item. Due to the colour-coding each participant already knew the parent item (context) of each target item and then only needed to visually scan each coloured context for the target item among other distractor items.

The tree structure on the right was made up of one of the three categories: Food, Animals and Countries. These three categories were believed to be commonly-known and were each easily divided into 6 sub-categories:

- Food: Alcohol, Candy, Dairy, Fruit, Meat, Vegetable
- Animals: Bird, Crustaceans, Fish, Insect, Mammal, Reptile
- Countries: Africa, Asia, Europe, North America, Oceania, South America

Each sub-category (context) consisted of 8 items, so that visual search at each context was done under the same conditions.

After clicking on the Start-Button, the participant selected the required target item(s) in the treebrowser, performed the copy operation, navigated to the textfield and pasted the items there. The participant repeated this procedure until the destination textfield contained all target items. Then the so far disabled Continue-Button was enabled and the clock for the trial was stopped (Figure 4.2). At this point the completed trial's data was analysed and logged and the participant was allowed to take a break if needed. By clicking on the Continue-Button the participant was able to move on to the next trial.

In total there were 18 pre-defined tasks, that were used across all three trial groups. Pre-defined means that target items for a specific trial were only semi-randomly generated. As not to be favourable to any particular technique it would be desirable to have

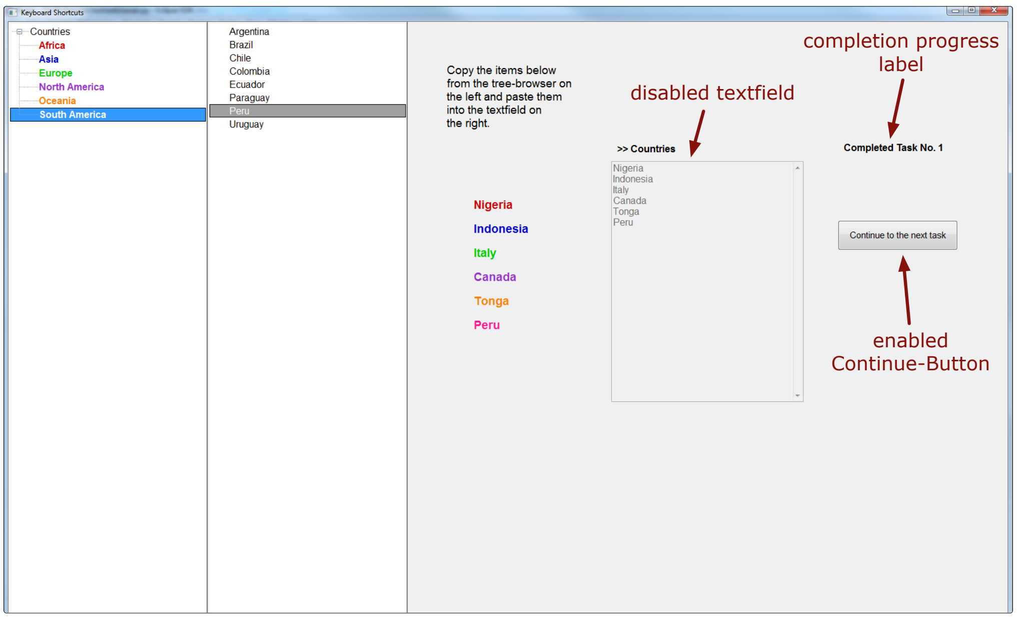

Copy the items below from the tree-browser on the left and paste them into the textfield on the right.

Figure 4.2: Screenshot of the experiment after completion of the first trial.

the same task for one specific trial across all techniques. The same target items should be chosen over all three techniques, since according to Fitts' law selection times for an item at the top of the list would be different to an item at the bottom of the list. Also the top item and the bottom item would differ in visual search times. However, using the exact same target items for one trial over all three techniques, would have made it very likely that participants would have been able to remember the location of these items' after having performed that specific trial a couple of times. Consequently, when being able to recall the items' locations participants would have been faster in selecting these target items. "Hardcoding" each single task therefore fosters a learning effect. To avoid a learning effect but still not favour any technique, only the region where a target item was located was pre-defined. That means a list of 8 items was split in three equal regions (3 items/3 items/2 items). Pre-defining the region of a target item guaranteed a similar location of target items but avoided a learning effect. Furthermore, the hierarchy for each task was randomly set. Because of the colour-coding, semantic knowledge of one specific hierarchy would not influence the selection times.

The 18 pre-defined tasks consisted of six tasks of each of the three item modes. In the 6@1 item mode, the participant was required to copy six items from one single context per trial. In contrast in a trial of the 6@3 or 6@6 item mode the participants needed to copy-paste six items from three or six different contexts respectively. A context in this experimental design referred to an item (folder) that contained sub-items. For example in the "Countries"-hierarchy, a continent item like *Oceania* is the parent item of *Australia* and *New Zealand* while *Europe* is the parent item of *Germany* and *England*. If the task is to copy-paste Australia and New Zealand, the task would consist of two target items at **one single context**. If the task, however, is to copy-paste Australia, New Zealand and Germany, the three items would be from **two different contexts**.

Half of the 18 tasks were interrupted. To be able to compare interrupted and non-interrupted tasks, there were essentially only 9 distinct tasks. Each of these 9 tasks existed twice in the set of 18 pre-defined tasks, once with and once without interruption. In an interrupted task, participants got interrupted after selecting one of the target items. When interrupted, the whole screen was overlayed by a grey frame obscuring the tree-browser, task description as well as the destination textfield. The grey frame displayed a mathematical problem that the participant was required to solve before being able to continue with the interrupted copy-paste trial. The mathematical problem consisted of a multiplication of two randomly generated numbers between 10 and 20 that then had to be added to another number (10-50). A calculator (Casio fx-82MS) was provided to support participants in solving the mathematical problems. Participants needed to put the problem's result into the textfield and click the OK-Button (Figure 4.3). If the result was correct and only then, the overlayed grey interruption frame disappeared so that participants could continue with the interrupted copy-paste trial from exactly the point they were interrupted.

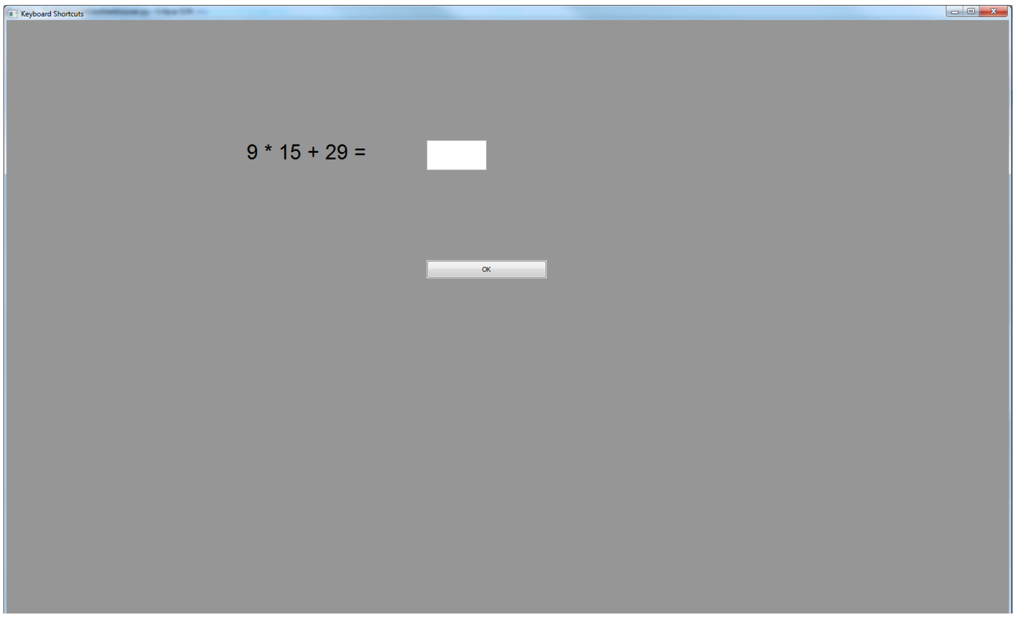


Figure 4.3: Interruption screen overlay with mathematical problem.

4.5 Treatment and Training

Participant completed the experiment by themselves and one after the other. To begin with, the participants received some general information about the experiment and were required to sign a consent form. Then they were asked to complete a small copy-paste task. They were supposed to copy all PDF files from a folder A on the Desktop into a folder B using their preferred copy-paste method. This was followed by a short interview to capture more about the participants copy-and-paste habits. Afterwards the general design of the experiment was explained to them.

Each trial group started with a training period used to explain the technique and the different types of tasks. Participants were allowed to train as long as they wanted to "until they feel at ease with the technique". The first trial started as soon as the Start-Button was pressed. At the start of a task the Continue-Button was disabled. Not Enabling the Continue-Button until all target items were pasted into the textfield prevented participants from accidentally clicking on the button and creating error-trials. To ensure participants were aware of their progress as soon as the Continue-Button was enabled, the destination textfield was disabled and a text message stating the completion of the trial was displayed (Figure 4.2). Participants were instructed to "perform as fast as possible without making any errors". In addition, since one could only advance to the next trial after having pasted all target items into the textfield, users were motivated to perform well. Even though all the target items had to be in the textfield, the textfield could still contain other items since items that were pasted into the textfield falsely could not be corrected or deleted. These other items (non-target items) were considered errors. Also, if target items were contained twice or even more often, they were also considered errors. A trial group was finished as soon as all 18 trials were completed. Each trial group was followed by a questionnaire, where participants were asked to rate the used technique according to different NASA-TLX questions [Hart and Staveland, 1988] using a 5-point Likert-scale.

4.6 Results

4.6.1 Empirical Results

The experiment took approximately forty minutes per participant. The experiment's log data was analysed using a repeated measures analysis of variance (RM-ANOVA). Performing an outlier analysis (trial time ± 3 s.d. from mean) on the completion time data detected no outliers. No trials were aborted. Three participants failed to adhere to the experiment's instructions (discussed in more detail in Section 4.6.1.2), hence their data was excluded. A total of 1026 trials (from 1188) were analysed (19 (of 22) participants $\times$ 3 copy-and-paste techniques $\times$ 3 items modes $\times$ 2 interruption conditions $\times$ 3 repetitions).

4.6.1.1 Completion Time

Completion time for one trial is the main dependent measure, and is defined as the time taken from clicking on the Start/Continue-Button until the destination textfield contained all required items. In case of an interruption, the clock was paused. Then as soon as the participant had solved the mathematical problem of the interruption screen correctly, the clock resumed immediately.

Q1. Is BND faster than KEY and DND? The mean completion time for a trial over all techniques, item modes and interruption conditions was 17.65 seconds with a standard deviation of 8.07 seconds. The three different copy-and-paste techniques reveal a significant main effect ($F_{2,36}$= 20.58, $p < 0.0001$) with DND fastest (Mean: 15.67, SD: 6.2), followed by BND (Mean: 17.75, SD: 7.4) and KEY (Mean: 19.55, SD: 9.8) (Figure 4.4). The means for the different item modes were 11.6, 16.9 and 24.5 seconds for 6@1, 6@3 and 6@6 respectively which provides a significant effect for item mode ($F_{2,36}$= 135.06, $p < 0.0001$).

As Figure 4.5 suggests there is also a significant *technique* $\times$ *mode* interaction ($F_{4,72}$= 18.86, $p < 0.0001$). Mean trial completion times increased as the number of contexts increased for all three techniques. DND's and BND's average trial completion

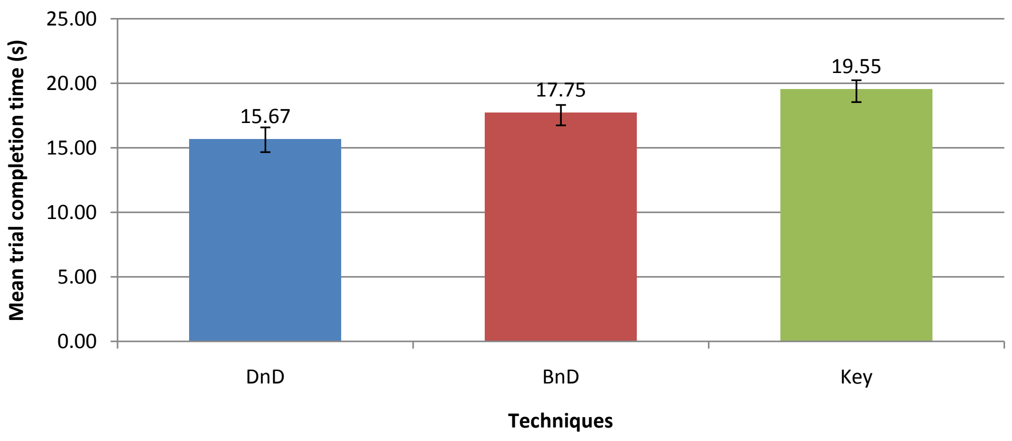


Figure 4.4: Mean trial completion times with DnD, BnD, and Key across all conditions. Error bars ±1 s.e.m.

times gradually (almost linearly) increased from 6@1 over 6@3 to 6@6 item mode. DnD's completion time was on average around 2 seconds quicker over all three items modes than BnD's. In contrast, Key's completion time taken for 6@6 increased dramatically from that for 6@1 and 6@3, and was on average more than 6 seconds slower than the other two techniques at 6@6. However, Key's mean completion time at 6@1 was similar to DnD's, and therefore slightly faster than BnD's completion time at 6@1. This suggests Key takes a bigger dip in performance the more contexts there are.

Q2. What effect do interruptions have on BnD's performance? As expected, all three techniques' average trial completion times were slower when interrupted. This provides a significant difference for the two interruption conditions with a mean for non-interrupted trials of 16.2 seconds and 19.2 seconds for interrupted trials ($F_{1,18}$= 68.61, $p < 0.0001$). There is a marginal interaction between *technique* and *interruption* ($F_{2,36}$= 3.249, $p = 0.05$). In Figure 4.6, BnD's slope is slightly steeper than the other

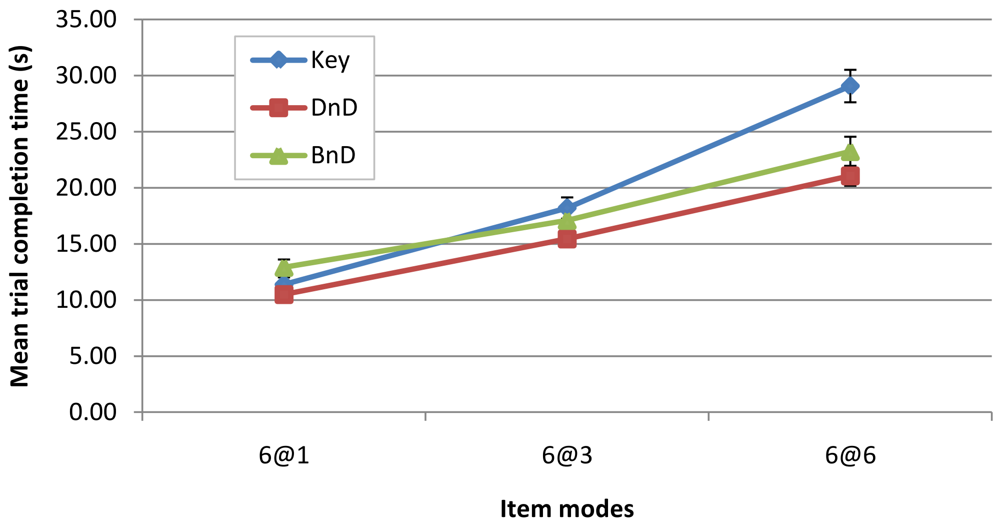


Figure 4.5: Mean trial completion times for DND, BND, and KEY over the three different items modes. Error bars ±1 s.e.m.

ones and therefore it might indicate that BND's rate of degradation in performance in terms of interruptions compared to the other two techniques might be greater.

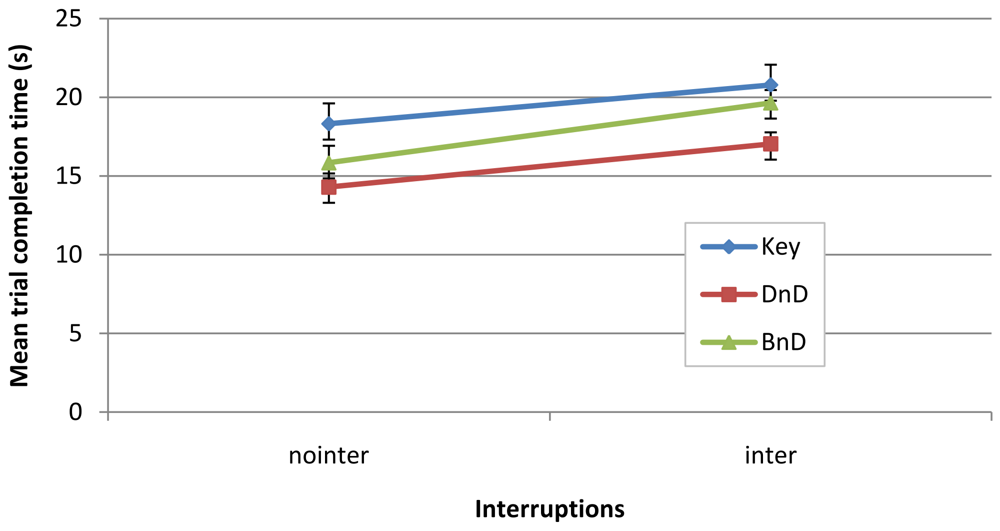


Figure 4.6: Mean trial completion times for DND, BND, and KEY with and without interruptions. Error bars ±1 s.e.m.

There are no significant interactions in *mode* × *interruption* ($F_{2,36}$= 0.47, p = 0.631) nor in *technique* × *mode* × *interruption* ($F_{4,72}$= 0.41, p = 0.801).

4.6.1.2 Error

As mentioned before, to complete a trial the textfield had to contain all required target items. Only then the Continue-Button was enabled and the participant could proceed to the next trial. As such, all trials are ultimately "successful". However, cases in which the textfield contained non-target items were classified as errors. In addition, the textfield could also contain certain target items multiple times. Such duplicate items were also regarded as errors. As mentioned before, the data of three participants was discarded. These three participants consistently made 2 errors in every 6@1 item mode condition across all techniques. At closer inspection, it turned out instead of following the task instructions and only copy-pasting the six required target items, they copy-pasted the six target items as well as the two remaining distractor items into the destination textfield. They therefore were not only outliers in terms of their error rate they also used a strategy that potentially could have been advantageous to them in terms of completion times. More to the selection strategies used in Section 4.6.1.4.

Q3. Are fewer errors made with BND than with KEY and DND? Overall participants made only 35 errors (Mean: 0.03, SD: 0.24), whereas 15 of them were duplicate errors. Even though hardly any errors were made, there is a significant main effect for technique ($F_{2,36}$= 3.39, $p < 0.05$). Users made the fewest errors with DND (Mean: 0.012, SD: 0.06), which was only slightly more accurate than BND (Mean: 0.017, SD: 0.09). Most errors were made with KEY (Mean: 0.073, SD: 0.27). Figure 4.7 shows the average number of errors per trial with and without duplicate items.

There is no main effect for item mode ($F_{2,36}$= 0.6, $p = 0.557$). It seems, though, there was a slight tendency that participants were more accurate as the number of contexts

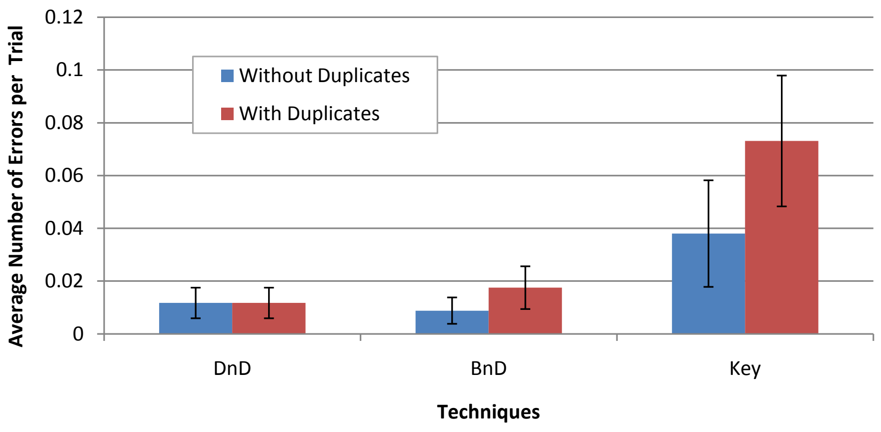


Figure 4.7: Average number of errors and duplicate errors per trial for DND, BND, and KEY. Error bars ±1 s.e.m.

increased with an average of 0.05 errors (SD: 0.2) for 6@1, 0.03 errors (SD: 0.2) for 6@3 and 0.02 errors (SD: 0.1) for 6@6. A higher error rate at 6@1 could be due to different strategies being used to select six items at one context (6@1), which is described in more detail in Section 4.6.1.4.

It also seems that as the number of contexts increased, the average number of errors decreased but the average number of duplicate errors increased. This seems to be particularly the case for KEY (Figure 4.8), even though there is no significant interaction between *technique* × *mode* ($F_{4,76} = 0.46$, $p = 0.767$).

There is no evidence of a main effect for interruption ($F_{1,18}= 0.38$, $p = 0.543$) nor for an interaction between *technique* × *interruption* ($F_{2,36} = 2.68$, $p = 0.082$). This is interesting, since interruptions only seem to have a significant effect on the completion time but not on the error rate.

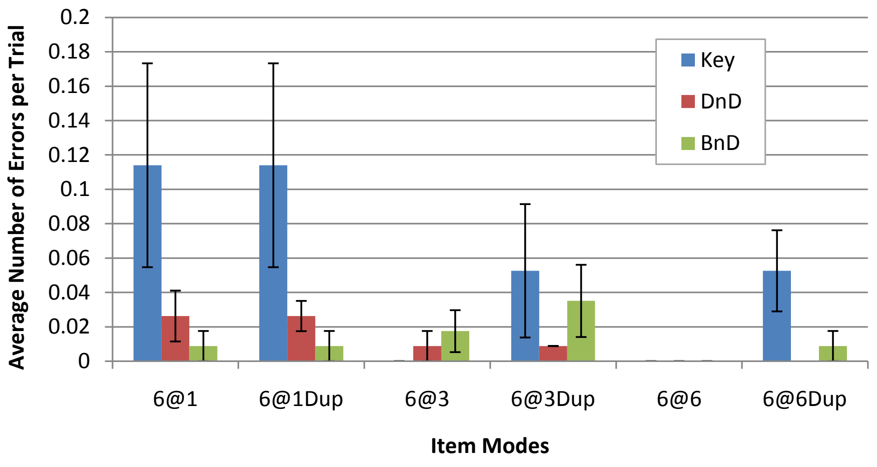


Figure 4.8: Average number of errors and duplicate errors per trial for DND, BND, and KEY at each item mode.

4.6.1.3 Mouse Travel Distance

Q4. Does BND reduce mouse travel distances? The mouse movement of one trial is recorded from the point the participants clicked on the Start/Continue-Button until the textfield contained all required target items. The travelled distance is measured in pixels. The average distance a participant travelled with the mouse to complete a trial was 7716.44 pixels (SD: 4459.52). As expected, there is a significant main effect for technique ($F_{2,36}$ = 3731.9, $p < 0.0001$). The average distance the participants had to move the mouse cursor to complete a trial with BND was 5567.13 pixels (SD: 2036.71) (Figure 4.9). In comparison using DND and KEY, participants on average had to travel much greater distances (almost twice as much) with 8433.82 pixels (SD: 4917.77) and 9148.37 pixels (SD: 4943.0) respectively.

The logged data also reveals significant main effects for item mode ($F_{2,36}$ = 3644.26, $p < 0.0001$) and interruption ($F_{1,18}$ = 48.6, $p < 0.0001$). Since participants had to go back and forth between one, three or six source folders and the destination textfield, the

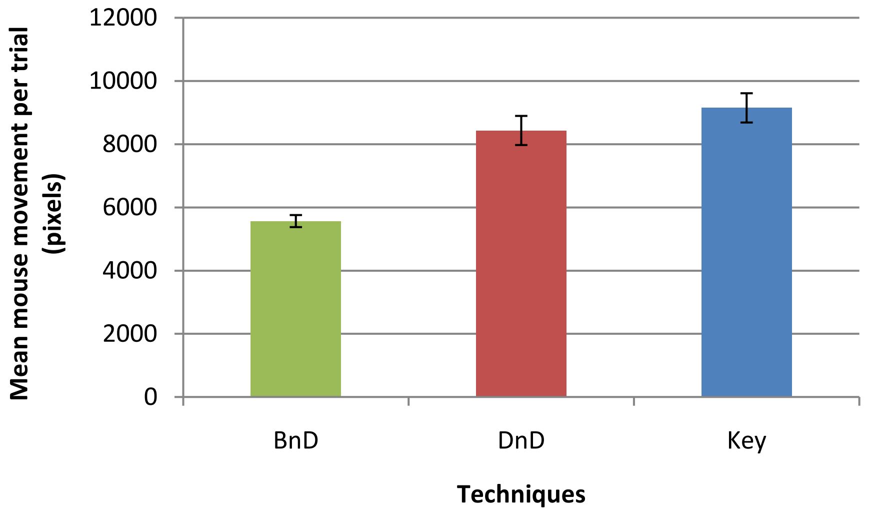


Figure 4.9: Mean mouse movement of one trial for DND, BND, and KEY. Error bars ± 1 s.e.m.

main effect for item modes is an expected result. The significant difference between the two interruption conditions could be caused by the additional distance participants had to travel from the location of the OK-Button of the interruption back to the tree-browser (in case they used the mouse and not the Enter-key to confirm the interruption screen's mathematical result). Also while re-orientating as caused by the interruptions, participants tend to move the mouse cursor around, which constitutes additional mouse movement.

Another interesting observation is that there is a significant *technique* $\times$ *mode* interaction ($F_{4,72} = 455.26$, $p < 0.0001$). The different slopes of the curves in Figure 4.10 illustrate that participants' average mouse movement with KEY and DND increased almost linearly with the number of contexts. While BND's average travel distance at one context (6@1) was slightly longer, participants had to move drastically shorter distances at three (6@3) and six contexts (6@6) in comparison to the other two techniques.

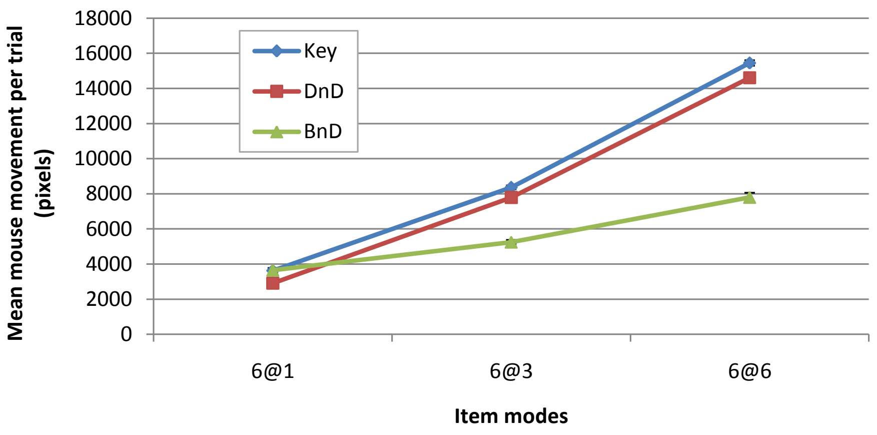


Figure 4.10: Mean mouse movement of one trial for DND, BND, and KEY at different items modes.

There was also a significant *technique* × *interruption* interaction ($F_{2,36} = 6.54$, $p < 0.005$). Figure 4.11 shows that interruptions have the smallest affect on KEY and the biggest on DND.

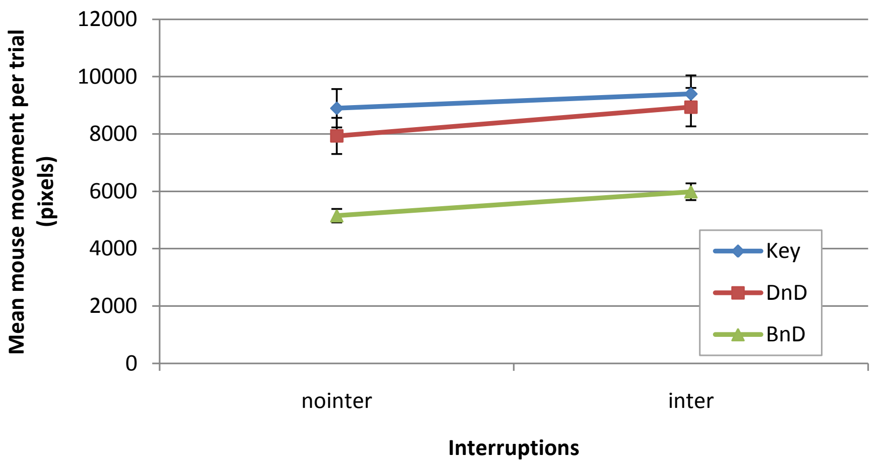


Figure 4.11: Mean mouse movement of one trial for DND, BND, and KEY with and without interruptions.

4.6.1.4 Selection Modes

When selecting multiple items at one context, participants could use three different selection modes pressing either Ctrl, Shift or a combination of Ctrl+Shift as described before. When using BND, participants were encouraged to use the Alt selection mode to select items across multiple contexts.

	6@1			**6@3**			**6@6**		
	Ctrl	Shift	Alt	Ctrl	Shift	Alt	Ctrl	Shift	Alt
KEY	96%	25%		98%	6%		35%		
DND	100%	22%		97%	8%		10%		
BND	16%	20%	84%			100%			100%

Table 4.1: Percental usage of selection modes in trials across different item modes and techniques (empty cells are equivalent to 0%).

Table 4.1 shows the percentage use of selection modes for each technique and item mode, revealing that participants often used Ctrl and Shift in the same trial. However, it also exposed that participants never held down Ctrl and Shift simultaneously as done for the additive range selection mode (Ctrl+Shift). Instead, they often ended up combining Ctrl and Shift successively, for example, by first selecting a few adjacent items using Shift and then adding the remaining non-adjacent items to the selection set using Ctrl. This is particularly the case in trials of the 6@1 item mode. As the results indicate, in every single 6@1 DND trial (100%) participants used the Ctrl mode, sometimes in combination with Shift. In contrast with KEY at 6@1 there were also trials in which participants exclusively used Shift to select the required items. Likewise, results for the 6@3 item mode show high use of the Ctrl mode for KEY and DND and only marginal use for the Shift mode. This is as expected, since there is no real benefit of using the range selection mode (Shift) when there are only two items to select at the time. Reason being, that regardless of using Shift or Ctrl to select two adjacent items you end up clicking on both items anyway, but to be able to select two non-adjacent items you have to use Ctrl. Furthermore, it is interesting to see that in 6@6 trials some participants held down Ctrl both for KEY and DND, even though there is no reason in doing so when only selecting one item at the time.

In contrast to that, with BND participants used Alt in 100% of the 6@3 and 6@6 trials, because the Alt selection mode was the only way to be able to select items across contexts. Likewise, in a majority of 6@1 trials participants used Alt. Only in a small number of cases they used Shift, Ctrl and/or a combination of all three. This seems logical, since in 6@1 trials participants only had to select from one context and therefore the ability of selecting from different context was not needed.

Participants used two different strategies to select the six target items at one context (6@1). Most users just selected each of the six target items by itself, pressing Ctrl or Atl to add the clicked-on item to the selection set. Several other participants first selected all eight items of one context using the Shift selection mode and then de-selected the non-target items. As mentioned before, three participants' data was discarded because they consistently copy-pasted all 8 items of one context (including the 6 target and 2 distractor items). This resulted in a high error ratio for the data of these participants at 6@1, meaning their data had to be classified as outliers. Their data revealed that two of the three actually used the latter selecting strategy pressing Shift to select all 8 items at once. However, they did not de-select the two distractor items, which inadvertently is faster. In contrast the third one actually selected each single item by itself using Ctrl, which is slower but cognitive more lightweight because he did not have to visually/semantically select the items requested in the task description. In the end all three participants did not adhere the experiment's instructions by deliberately making mistakes and therefore their data was excluded.

While pressing one of the modifier keys Ctrl, Shift and Alt, clicking into empty space had no effect on the selection set. However, releasing the modifier keys and then clicking into empty space resulted in all selected items being de-selected, in other words the selection was lost. With BND, selection was lost in 3 out of 342 trials ($<1\%$). Hereby, one of these three trials was at 6@3 and two at 6@6, whereas two of them were interrupted trials. In comparison with DND and KEY the selection was lost each in one trial (both interrupted).

In terms of selecting items, another interesting observation is the time it took participants to click on, that is to select, an item in the treebrowser after first starting a trial. Participants were fastest when using DND with 1.67 seconds (SD: 0,6), followed by KEY with 1.71 seconds (SD: 0.5) and then BND with 1.91 seconds (SD: 0.7), which provides a main effect for technique ($F_{1,18} = 5.92$, $p < 0.01$).

4.6.1.5 Bi-Manual Interaction

To select multiple items, participants were encouraged to use multiple item selection modes by pressing modifier keys like Ctrl, Alt and Shift with one hand while at the same time point and click with the mouse using their other hand on the desired item. Only in such a bi-manual way could multiple items be selected. In addition, executing copy and paste commands for the KEY and BND copy-paste techniques, participants were required to evoke keyboard shortcuts. Most participants therefore used their left hand to interact with the keyboard while placing their right hand on the mouse to control the pointing. In this sense, participants could move the mouse cursor while at the same time they were activating the copy/paste commands via keyboard shortcuts.

As expected participants executed Ctrl-C/V in average 3.95 times per trial with KEY (SD: 2.08) and only 1.02 times per trial with BND (SD: 0.14). To see if there was a difference in times lost by pressing keyboard shortcuts, I performed an ANOVA similar to the main ANOVA but only with BND and KEY. The main measures were the time and the travelled distance from the moment participants finished the selection of one or multiple items to the moment they pressed Ctrl-C to copy the selected items to the clipboard. The average time was 981ms with a standard deviation of 416ms. The average times of 813ms for KEY and 1149ms for BND provide a main effect for technique ($F_{1,18}$ = 86.13, $p < 0.0001$). There is also a main effect for interruption ($F_{1,18}$ = 88.86, $p < 0.0001$), with 866ms for non-interrupted trials and 1096 for interrupted trials. During this time, participants moved the mouse on average 499.55 pixels (SD: 270.77 pixels), which shows no significant main effect for techniques KEY (Mean: 484 pixels) and BND (Mean: 515 pixels) ($F_{1,18}$ = 2.12, $p = 0.16$). For some participants though, the distance they travelled was as short as 0 pixels and for others around 2500 pixels. This shows that participants followed different strategies. Furthermore some of the participants used two hands to evoke keyboard shortcuts. This explains why some did not move the mouse after selecting the last item until pressing the keyboard shortcut.

The average time it took participants from the moment they placed the cursor in the destination textfield to the moment they pressed Ctrl-V was 520ms (SD: 317ms). This is a fairly big value considering it is only one mouse click plus the activation of one keyboard shortcut. For some participants, however, it went down to close to 100ms.

4.6.2 Qualitative Results

Q5. What is the subjective response to BND? Overall participants responded positively to BND and also provided many suggestions for improvements. After completing trialgroups with all three different copy-and-paste techniques (KEY, DND, BND), participants were asked to rank them in order of preference, with 1 most preferred, and 3 least. Fourteen ranked BND first, three KEY, and one DND. This provides a significant rankings difference (Friedman χ^2_r = 11.73, $p < 0.005$) with means of 1.41 (SD: 0.59), 2.23 (SD: 0.69), 2.36 (SD: 0.85) for BND, DND and KEY.

	KEY	**DND**	**BND**	χ^2_r	**Sig**
Metal demand	2.9 (1.4)	2.4 (1.3)	2.6 (1.4)	4.0	=0.13
Physical demand	3.2 (1.5)	2.8 (1.3)	2.1 (1.1)	7.6	<0.05
Successful	4.1 (0.8)	4.3 (0.6)	4.2 (0.8)	0.4	=0.80
How hard	3.0 (1.3)	2.6 (1.0)	2.3 (1.1)	7.0	<0.05
Annoyed	2.9 (1.6)	2.3 (1.2)	2.0 (1.1)	6.3	<0.05
Efficient	3.5 (1.0)	3.9 (1.0)	4.4 (0.9)	12.1	<0.01
Easy-to-learn	4.5 (0.6)	4.5 (0.5)	4.4 (0.9)	0.1	=0.97
Interruption	3.4 (0.8)	3.4 (1.3)	4.1 (1.3)	4.5	=0.11
Error prone	2.5 (1.3)	2.3 (1.1)	2.0 (1.0)	2.8	=0.24

Table 4.2: Mean responses (Standard Deviation) to 5 point Likert-scale questions.

After each trialgroup participants were asked to rate the technique they just used according to different NASA-TLX worksheet questions using a 5-point Likert-scale. The responses are summarised in Table 4.2. The results show significant differences for physical demand, amount of work and annoyance. Participants perceived BND as physically lower demanding than the other techniques. They also felt it was easier to accomplish the level of performance with BND than it was with KEY or DND. Table 4.2 also shows that participants rated BND's efficiency (Mean: 4.4) significantly higher than DND's (Mean: 3.9) or KEY's (Mean: 3.5). Not significantly different but interesting is that mean ratings for BND's ease of learning (Mean: 4.4) were just slightly better than the others (Mean: 4.5), despite the very brief exposure participants had to BND. Also participants seemed to feel slightly better supported by BND in case of interruptions and believed to be more accurate with BND.

I also measured how visually distracting participants found the bag-like visual feedback of BND. Turns out participants did not find the bag (bubble) distracting. One participant stated that he found it *"awesome"* and another that he *"really liked the bubble"* but he suggests that *"the bubble should remain visible even after pressing Ctrl-C"*. The mean response to the question "BND's feedback was visually distracting" (1 disagree, 5 agree) was 1.66 (SD: 1.25). Furthermore, 18 participants rated "BND's ability to copy-paste across contexts" as very useful (Mean: 4.82, SD: 0.70). Participants stated they found BND "clever", "handy" and "very useful". Only one stated, he thinks BND would not be that useful to him because he does not "often copy across contexts". Furthermore, most participants perceived BND as "fun to use" (Mean: 4.5, SD: 0.78) and they would like to use BND on their own computer (Mean: 4.5, SD: 0.96). One participants explicitly said he found BND "not only fun to use but also really enjoyed its pretty Mac-like appearance".

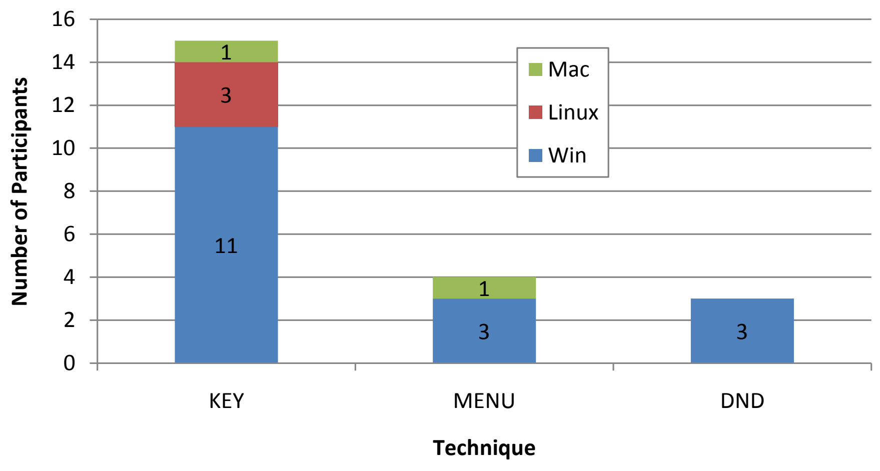
4.6.3 Copy-and-Paste Practices

Before starting with the training tasks for the experiment I interviewed the twenty-two participants on their copy-and-paste habits. As mentioned before, I considered the twenty-two participants as "expert users", all of them being postgraduate Computer Science students. Before asking specific questions on copy-paste habits, the participants were asked to complete a small copy-paste task. Participants were asked to copy all

PDF files from folder A and paste them into folder B using their preferred copy-paste method. Both folders were located on the Desktop. The source folder A contained 5 PDF files and 13 distractor files of other filetypes (jpg, doc, exe).

All twenty-two interviewed people said they use copy-paste very often. To accomplish the small copy-paste task, seven Windows users used keyboard shortcuts Ctrl-C/Ctrl-V to copy-paste all PDF files into the destination folder. Three Windows users chose to use the context menu (right-click menu) and five others used drag-and-drop. There were also 2 Windows users that used a combination of keyboard shortcuts and menu. They copied all selected PDF files to the clipboard via Ctrl-C and then right-clicked onto the destination folder and executed the "Paste into" command to paste the clipboard's content into the destination folder without even having to open the folder itself. Two of the Linux users also chose to use the keyboard shortcuts and menu combination while one Linux user stuck with the traditional Ctrl-C/V. The two OS X users were divided in one using the context menu and the other one using drag-and-drop.

These observations from the small copy-paste task slightly differ from what the participants stated when asked directly about their copy-paste habits (Figure 4.12). In the interview, 15 participants said they prefer to use to keyboard shortcuts (Windows: 11, Linux: 3, Mac: 1). Six of them reasoned that keyboard shortcuts are just the easiest to use. Another six said it is convenient to use them because they have the left hand on the keyboard anyway and thus they can use both hands in a bi-manual way. Several stated keyboard shortcuts are just the fastest way to copy-paste. And a few explained they are just used to it. Only four participants stated they use menu copy-paste (Windows: 3, Mac: 1) more than keyboard shortcuts or drag-and-drop. Among these four users of Menu copy-paste, two mentioned they use it out of “laziness” because they only need to use one hand. Three participants, all of them Windows users, said they usually use



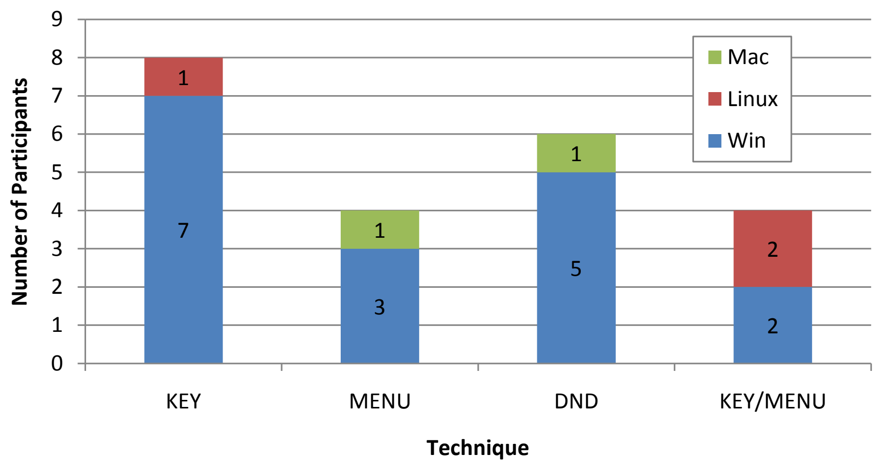


Figure 4.12: The top graph shows the preference of copy-paste techniques stated by the participants in the interview, while the bottom graph shows the techniques used in the copy-paste task.

drag-and-drop. One of them explained in the interview that he has two large screens and can therefore easily arrange two windows (folders) next to each other and copy-paste quickly between them.

All five participants that used drag-and-drop to complete the copy-paste task moved the PDF files into the destination folder B instead of copying them. When asked about drag-and-drop's cut-and-paste/copy-and-paste ambiguousity, four of them were unable to say if a drag-and-drop moves or copies the selected items. After explaining the ambiguousity to them, most said they had run into this problem. Only one of the participants was aware that holding down Ctrl under Windows while dragging items, copies the dragged items instead of moving them.

To copy-paste the PDF files participants were obviously first required to select the PDF files. Most participants either used the Shift or the Ctrl selection mode. That means they either selected each file individually while pressing Ctrl to add them to the selection set or they ordered the files in the source folder A by file type and then selected the range of all PDF files by pressing Shift. However, there were three participants that did not know how to use the Ctrl nor Shift selection mode. They were trying to select the files individually but instead of pressing Ctrl, they were holding down Shift, they then got confused by the unexpected range selection of files on the screen. In the end these three participants copied the files one by one.

I also asked a few specific questions about clipboard extensions such as the Office Clipboard introduced in Chapter 2.4.1. Several participants said they have seen the Office Clipboard before but they do not use it. Only one participant confirmed using the Office Clipboard from time to time. And one other participant stated to be using Ditto quite frequently.

CHAPTER 5

Discussion and Future Work

5.1 Summary of Findings

The results from the experiment show that participants were fastest when using Drag-and-Drop (DND), even though the majority of participants preferred Bag-and-Dump (BND). In tasks with one, three and six contexts (folders) as well as in interrupted and non-interrupted ones, DND was significantly faster than BND and Keyboard Shortcuts (KEY). BND in turn was significantly faster than KEY. Regardless of the technique being used, participants' average completion times increased relative to the number of contexts. Compared to BND and DND, KEY had a much larger degradation in performance with the increase in number of contexts. Participants were significantly slower in interrupted tasks, although there was no significant difference among techniques. Participants made the most errors with KEY and the least with DND. BND's mouse movement was substantially shorter than other techniques.

5.2 Experimental Concerns

5.2.1 Inconsistency of DnD

The result of a drag-and-drop operation can be quite ambiguous. In some situations a drag-and-drop action copies items to the destination, leaving the original object unaltered at the source, whereas in another situation items are moved. In the Windows Explorer, drag-and-dropped items are always moved as long as they are on the same medium (harddisk). For example drag-and-dropping items from a flashdrive onto your harddisk would copy the items, leaving the original files behind on the external flashdrive. In general though, to be 100% sure that items are only copied and not moved the user has to hold down Ctrl while dragging. Even though participants were asked to copy items in the experiment, it was not required of them to hold down Ctrl. This depicts a slight deviation from the original Windows Explorer drag-and-drop implementation. Continuously holding down Ctrl while dragging constitutes an additional low level interaction in the form of continuous muscle tension that could increase physical strain as well as cognitive load. Therefore, the necessity of holding down Ctrl while dragging could have had a negative effect on the performance of DnD and could therefore be regarded as a potential threat to the validity of the results of the experiment.

5.2.2 Validity of Interruptions

A potential confounding factor could have been the random determination of the exact moment of interruptions. In half of the trials of the experiment participants were interrupted while selecting target items. The exact moment of interruption was randomly determined for each trial and occurred after the participant had selected between 2 to 5 items. So the questions is; did interruptions after the second selection have the same effect as interruptions after selecting the fifth item? It seems, not controlling the exact moment of interruption and not counter balancing it across techniques and participants could have been a confounding factor. However, trial completion times in regard to the interruption point in time do not show any significant difference.

Furthermore, the validity of only using interruptions while selecting is questionable, since in natural copy-and-paste tasks interruptions can occur at any stage of a copy-paste process. As explained in a small scenario in Section 3.5.2, interruptions before pasting, for example, can have a significant effect on the copy-paste workflow. The reason for deciding on interruptions while selecting was that dragging operations cannot be interrupted without being completely aborted. The aim of the experiment was to compare DND and KEY, two of the most used standard copy-paste techniques, with BND. To be able to compare them, participants have to do the same kind of tasks with all three techniques. In order to incorporate DND in the experiment it was necessary to restrict interruptions to the process of selecting. However, due to this restriction the experiment may not be considered as a full investigation into the effect of interruptions.

5.2.3 Inconsistency in the experiment's user interface design

The experiment was based on file manipulation using copy-and-paste in a standard file browser. The treebrowser widget used to conduct the test was closely designed (in terms of appearance and behaviour) on the basis of the Windows Explorer and the standard techniques, KEY and DND, were supposed to behave as they usually would under Windows. However, since the experiment's user interface might have behaved slightly different in terms of selection mode behaviour than the Windows Explorer, there could have been a potential confounding factor.

For example, if a user selects a range of items by pressing Shift, but then decides (s)he wants to de-select a few of the just selected items in the middle, (s)he can do so by clicking on them while holding down Ctrl at the same time. Now if (s)he ends up wanting to re-select all or some of the de-selected items in the middle, you would expect that (s)he is able to do so by holding down Ctrl+Shift and clicking onto the top or bottom unselected item. This should add the range of unselected items in the middle to the currently selected items. However, under Windows this is not the case. The user is not able to re-select any of the middle items by using the additive range selection mode (Ctrl+Shift). In fact holding down Ctrl+Shift and clicking onto items appears to have no effect whatsoever. To re-select these items, it seems the user has to add each of these items to the selection set by holding down Ctrl and clicking onto them individually, even though they are all adjacent items that (s)he should be able to add as a range. There are a few other scenarios along the same lines that exhibit similar random behaviour – in particular with the Ctrl+Shift selection mode. Since there is no logical way of knowing how Windows behaves, it is hard to simulate the behaviour of the treebrowser widget one-to-one. Therefore, the possibility of a small inconsistency to the selection mode behaviour in the Windows Explorer could be seen as a potential confounding factor. However, because none of the participants ended up using the Ctrl+Shift selection mode, this specific scenario did not occur but other similar inconsistencies might have.

5.3 What succeeded with BND?

BND was preferred by a majority (73%) of participants. BND's ability to carry selections across contexts was designed to overcome the limitations of current copy-paste techniques. The experiment showed that BND has the potential to do so, since it was 9% faster than KEY, which, according to the conducted interview as well as other studies [Tak, 2007; Chapuis and Roussel, 2007], is believed to be one of the most popular techniques among "expert users".

BND's performance compared to KEY was better particularly with the increase of contexts. The more contexts there were involved in the copy-paste tasks the bigger was the difference in distance the participants had to cover with KEY compared to BND. Essentially the bigger the difference in distance was the participants had to cover, the bigger was the time difference between average completion times of KEY and BND. The experiment not only confirmed that with BND mouse movement compared to KEY was almost cut in half, it also validated this in relation to DND. In addition to participants being faster with BND than with KEY they were also more accurate. Participants made a significantly smaller number of errors when using BND than with KEY.

Apart from being faster than KEY, BND was also the preferred technique. Most participants stated BND was "very useful". The NASA-TLX ratings showed that participants felt physically less exhausted using BND. This might be due to the significantly shorter distance participants ended up travelling to complete the trials. Possibly, due to the lower physical demand, participants perceived it to be easier to accomplish a high level of performance. The constant visual feedback and its visual appearance might have been another factor contributing to the perceived ease-of-use of BND. Results also showed a tendency that participants perceived BND easier-to-learn than the other techniques. This is an interesting finding considering participants only had a very brief exposure to BND compared to the long experience they have had with the other techniques. Participants' NASA-TLX ratings also confirmed that BND was significantly more efficient than the other techniques.

5.4 What did not work with BND?

Apart from confirming that BND was faster than KEY, the empirical results also showed that BND was about 11% (on average about 2 seconds) slower than DND. This means, even though with DND participants had to move the mouse almost twice as far as with BND, they were still faster with DND.

5.4.1 Participants' individual performance

The scatterplot in Figure 5.1 shows the average trial completion times of each individual participant for all three techniques as well as each participants' overall average trial completion time. It visually stands out that some participants have quite big gaps between their datapoints of their average completion times of different techniques, while

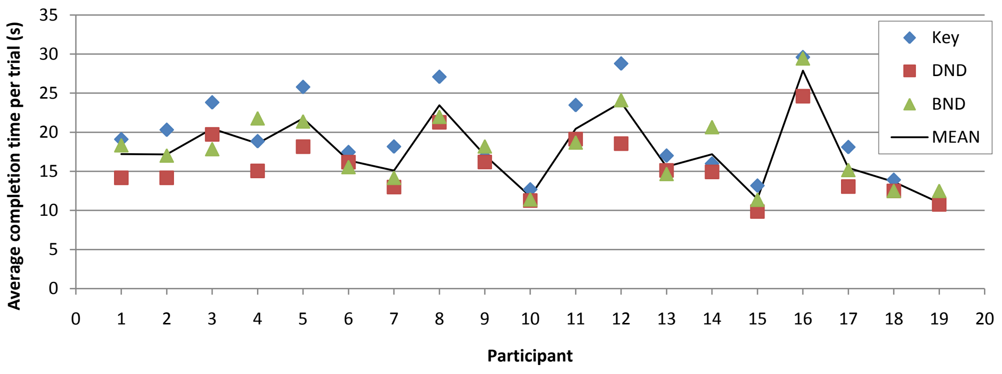


Figure 5.1: Average trial completion times of each participant.

for others the different technique's datapoints are very close together. In other words, participants with big gaps had a greater variance in performance across techniques. In order to have a closer look at individual performances, participants are divided into two groups:

- Participants with a *big gap*
- Participants with *no gap*

It appears that datapoints of different techniques for participants of the *no gap* group (6, 9, 10, 13, 15, 18, 19) are so close together that they are about to merge into one big data point. For these participants BND and DND trial completion times were very similar; sometimes BND was a little bit faster than DND or vice versa. For these participants, KEY was on average the slowest out of the three. Likewise, for all participants of the *big gap* group, KEY was on average by far the slowest technique. For the majority of the big gap group, DND was the fastest technique (on average at least 5 seconds quicker than KEY). Thus, the time difference between DND and KEY accounts for most of the big gaps in the scatterplot. For most of the *big gap* group, BND was somewhere in between KEY and DND.

Another interesting observation is that a big gap seems to be equivalent to slow completion times. Participants of the *no gap* group were on average overall about 6 seconds faster than participants of the *big gap* group. In addition, there is not one participant with a big gap, who performed faster than everyone else with one technique and slowest with another technique. If participants have big gaps then they also had slow average trial completion times. Plotting the time difference between the slowest and fastest technique's average completion time against the overall average trial completion time of each participant as done in Figure 5.2 confirms this observation.

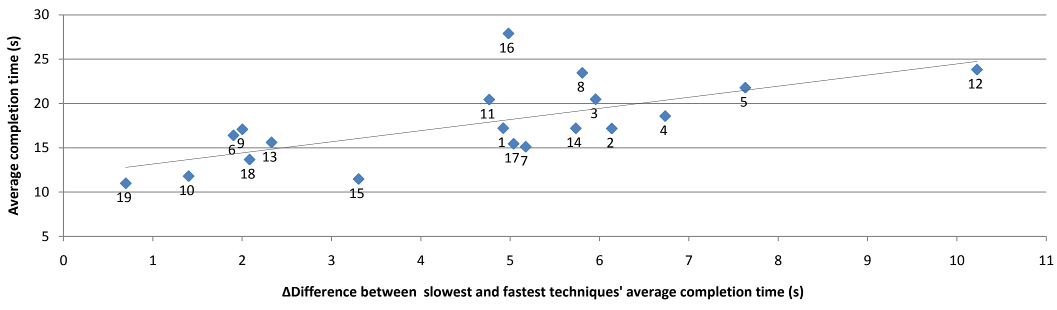


Figure 5.2: Each participants' time difference of average completion times across all three techniques compared with their overall average completion time.

The scatterplot of Figure 5.2 shows that variance correlates with performance. Previously exposed *no gap* participants are in the left bottom section of the graph, in the group with the lowest time difference between their slowest and fastest technique's average completion time. The scatterplot validates that these participants are also the participants with the fastest overall average completion times. The relationship between the overall performance (average completion time) and the variance of performance (time difference between the slowest and fastest technique's average completion time) was investigated using Pearson's product-moment correlation coefficient. Pearson's r-value (r = 0.68) shows there is a strong positive correlation between the two variables [Cohen, 2003]. To summarise, fast participants performed consistently fast across all three techniques, which means they had a low variance in performance, while slow participants' performance varied across techniques and they therefore had a high variance.

5.4.2 Why variance correlates with performance

This pattern raises the question, why do slow participants have a higher variance between their completion times? Or in other words, why were they not as good with BND and KEY as they were with DND? It seems there has to be a correlation between BND's and KEY's performance. If participants were slow with KEY (around 25 seconds), then they were also slower with BND than DND. However, if participants were fast with KEY (around 15 seconds) then they were definitely also fast with BND, possibly even faster than with DND. This suggests a common factor between KEY and BND that distinguishes them from DND. In terms of the low level interactions of these techniques, the big difference is the execution of the copy-paste commands. DND uses one single, continuous, direct manipulation drag interaction to execute the copy-paste commands. In contrast, KEY and BND make use of two keyboard shortcuts Ctrl-C and Ctrl-V to invoke the copy- and respectively paste-command. Since KEY and BND follow different concepts when it comes to selecting items, their obvious common denominator is the use of keyboard shortcuts Ctrl-C/Ctrl-V.

Overall performance of participants with different techniques correlates with their performance executing keyboard shortcuts. When executing keyboard shortcuts, bimanual interactions play a major role. As mentioned in the results section (Chapter 4.6.1.5), participants took on average 981ms to activate Ctrl-C and about 520ms to press Ctrl-V. This means performing these two keyboard shortcuts just once, accounts for more than 1.51 seconds. There were even some participants that took significantly

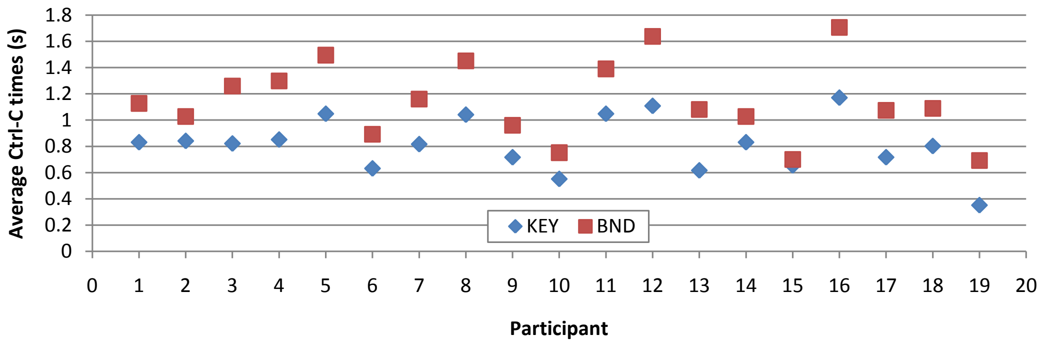


Figure 5.3: Average Ctrl-C times of each participant.

longer than this. Even though participants were instructed to use only one hand to execute keyboard shortcuts, a few participants (e.g. 12 and 16), even after being reminded to use only one hand, kept on slipping back into their routine of using two hands, holding down Ctrl with one finger of the left hand and pressing C or V with the index finger of the right hand. Because they had to move their whole arm from the mouse to the keyboard to press C/V, it took them considerably longer to execute Ctrl-C/V. On average participants moved the mouse cursor about 500 pixels while invoking Ctrl-C at the same time. However, participants that used two hands to invoke keyboard shortcuts were obviously not able to move the mouse at the same time and therefore not only took longer to press Ctrl-C but also ended up having the slowest average completion times. Figure 5.3 shows the average time it took each individual participant to execute Ctrl-C with KEY as well as with BND. Comparing Figure 5.3 and Figure 5.1 clearly indicates that the slowest participants in terms of average completion times were also the slowest performing keyboard shortcuts. Figure 5.4 shows a graph that maps the average execution time of the keyboard shortcut Ctrl-C to each participants' overall average completion time. Performing a correlation analysis on these two variables shows a strong positive relationship (r = 0.93). This confirms the speculation that the slowest participants had a harder time using keyboard shortcuts, which could explain why they were significantly slower with BND and KEY than with DND.

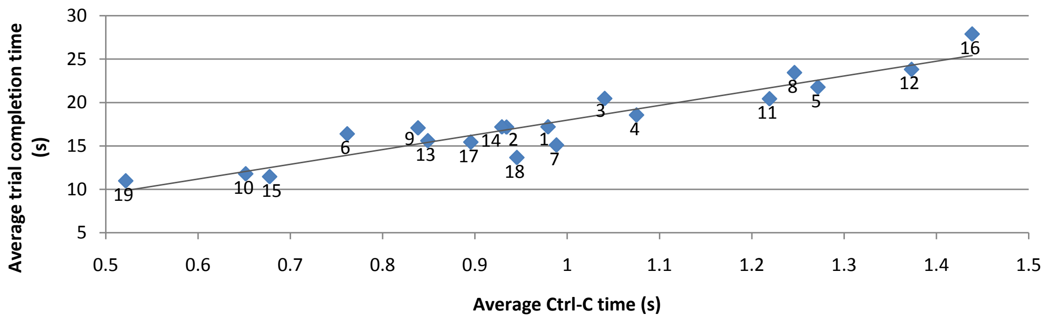


Figure 5.4: Each participants' average Ctrl-C time compared to their overall overall average completion time.

In general, BND requires only one execution of Ctrl-C/Ctrl-V. According to the experiment's data, participants invoked keyboard shortcuts Ctrl-C/V on average 3-times as often with KEY than with BND. Therefore, the performance of keyboard shortcuts had a bigger effect on the completion times when using KEY. Apart from the greater mouse distance participants had to cover with KEY, this adds another factor to KEY's overall slower performance in comparison to BND.

When looking at the bi-manual execution times of Ctrl-C, results show it took participants longer to press Ctrl-C when using BND. This might be due to the use of the Alt-key when selecting across different contexts with BND. Instead of holding down Ctrl or Shift to select multiple items, participants were required to hold down the Alt-key when using BND. They hereby obtained the ability to select items across different contexts. As soon as all items were selected they had to let go of Alt and press Ctrl-C to copy the selected items to the clipboard. The physical shift of fingers from the Alt- to the Ctrl-key required extra time. According to the usage of selection modes in Chapter 4.6.1.4, participants used Alt most of the time to select multiple items with BND. In contrast with KEY, participants almost exclusively used Ctrl to select multiple items. Since they then were already holding down the Ctrl-key, they did not have to shift fingers to invoke the Ctrl-C copy-command. This gives KEY an advantage when invoking keyboard shortcuts and consequently explains the slower BND Ctrl-C performance times.

The experiments results show that the Alt and Ctrl-C combination of BND was slow in comparison to KEY's standard combination. Participants were required to perform unfamiliar motor operations in terms of moving their finger from the Alt-key to Ctrl. Most participants, who are 10-finger typists, tend to leave their little finger either on the Shift or Ctrl key while resting their left hand on the keyboard. This also makes it easy for them to hold down the said modifier keys when selecting with conventional selection modes. In addition, it eventually allows an easy execution of Ctrl-C/V. In contrast, holding down Alt requires participants to either shift their little finger from the original location close to Shift/Ctrl to the Alt-key, or possibly even use another finger (such as the thumb), to hold down Alt while selecting with BND. Either way this requires a physical shift of the hand when pressing Ctrl-C/V to copy the selected objects to the clipboard. This hand shift constitutes additional costs. Therefore, the traditional combination of Ctrl/Shift and Ctrl-C seems not only easier but in particular quicker. This was validated by some of the participants' feedback. One participant stated, "if the keys were mapped better, BND would be much easier". Another suggested to use the Ctrl-key instead of Alt. One particular participant, a Mac user, noted that he quite often made the mistake of pressing Alt-C/V instead of Ctrl-C/V. This is not surprising, since Mac OS users are used to holding down the Alt-key while selecting and copying. Instead of pressing Ctrl, Mac OS users need to hold down the command-key to select multiple objects as well as press command-C to copy these selected objects to the clipboard. On a Mac's keyboard the command-key is located at the same position as the Alt-key on a PC's keyboard. This explains the mix-up of keys for Mac users.

5.4.3 Why participants were faster with DND

The big difference to KEY as well as BND – and as it turns out also the biggest advantage of DND – is that DND does not involve any keyboard shortcuts. Instead, to execute the copy-paste commands participants were only required to use a direct manipulation interaction solely based on mouse input. They only had to select the desired items, then, in one motion, click on the selected items and hold down the left mouse button, drag the mouse cursor attached items to the destination and release the mouse button to drop the items there. Dragging, when done over a long distance, is renowned for being error-prone and slower than normal pointing [MacKenzie et al., 1991]. Furthermore, while dragging a user cannot perform any additional auxiliary tasks. This means the source (window) and the destination have to be aligned and visible (that is, placed on top of all other windows) on the screen before the user even starts dragging the objects. However, over a short distance with a large drop-off area that is visibly arranged next to the source, DND has proven to be one of the easiest and fastest ways of performing copy-and-paste operations [Chapuis and Roussel, 2007].

Dragging items from the source and dropping them at the destination proved to be the fastest copy-and-paste technique, although the experimental design did not truly test its limitations. The experiment did not demand the execution of any auxiliary tasks since the source of the item (treebrowser) as well as the destination (textfield) were fixed to their locations on the screen and visible at all times. Participants only had to click onto the desired objects and keep on holding down the left mouse button to start the drag action. That means, in contrast to the other techniques they did not even have to release the mouse button. In a next step, they had to drop the dragged items – that is, release the held down mouse button over the textfield – whereas with the other techniques participants had to specifically click into the textfield to mark the insertion point. The textfield itself was quite big in relation to the screen and therefore an easy dropping target. In contrast, after clicking into the textfield with KEY and BND (which in terms of low level interaction is very similar to the dropping action) participants still had to perform the paste-command via the keyboard shortcut Ctrl-V. This means DND's amount of low level interactions – from the point the selection processed is finished, up to the point the items are pasted at the destination – is considerably smaller. This again seems to have a greater effect on slower participants.

5.4.4 Why BND's selection mode increased cognitive load

Since KEY and DND are based on the same selection mechanism, the smaller amount of low level interactions to execute the copy-paste commands combined with, in particular, slower participants' slow performance of keyboard shortcuts may explain the time difference between KEY and DND. BND, on the other hand, is based on a different selection mechanism; the new across context selection mode. As a result, BND's mouse movement and the amount of copy-paste commands executed was cut down considerably. Not having to travel from source to destination and back shortened mouse movement to almost half the distance. In addition, copy-paste commands needed only to be executed once per trial. Consequently, it seems the shorter distance should have outweighed the additional time it took participants to execute one copy-paste command via Keyboard Shortcuts and therefore should have resulted in faster completion times with BND compared to DND. In particular, faster participants, who evidently were less affected by the use of Keyboard Shortcuts, should have been quicker with BND than with the other techniques.

A possible explanation might be an increase in cognitive load due to the use of the across context selection mode, which led to a slower performance of participants. With KEY and DND a trial could basically be broken down into a set of small, individual copy-paste tasks at each context. Each of these individual tasks only required the selection of items at one context and then copy-paste them to the destination. This means, with KEY and DND one trial in the experiment consisted of up to 6 small, separate copy-paste interactions that could be performed independently – whereas with BND, one trial consisted of one big, combined copy-paste interaction. When using BND's across context selection mode, participants were required to select all target items before copy-pasting them. This more extensive selection process had to be treated as one continuous action, whereas the participants were required to keep track of their selection progress. Even though the 'bag' provided constant visual feedback of the selection task's progress, participants still needed to process this visual information and translate it into information enabling them to complete the task. Therefore, the selection task when selecting across contexts involved a larger amount of information that needed to be processed at the same time, which, in turn, resulted in a higher cognitive load.

Another spin-off of having a large selection task is the contrast between the reasonably demanding task of selecting all of the items and the disproportionally simple action (one accidental click would suffice) that would cause these items to be lost. While selecting multiple items, regardless of the selection mode, participants could lose the whole selection set due to its volatility (accidentally releasing the modifier key and clicking somewhere). Consequently, in an attempt to improve accuracy and avoid losing the selection set, participants performed slower. Since BND's selected items could consist of items from different contexts though, losing the selection set would constitute a bigger loss than losing the selection set with the other techniques. A loss with BND would imply participants having to go back to each single context and select the items there again. In contrast with KEY and DND, a selection set can only consist of items from one context and therefore rectifying a loss of the selection set would only involve re-selecting items at the last context. Thus the 'fear' of committing a mistake with BND that would result in the loss of the selection set while selecting across contexts might have been greater among participants and thus might have led to a slower (but more accurate and conscious) interaction process.

Evidence of BND's higher mental effort while selecting might be the slower start-selection times as well as the tendency of a higher degradation in BND's performance in interrupted trials. At the start of a trial it took participants significantly longer to click onto an item in the treebrowser when using BND. Similarly, in interrupted trials it took participants longer to continue the interrupted trial in relation to the other two techniques. This means the process of orienting at the start of a trial and the process of re-orienting after interruptions took considerably longer with BND, which may signify a higher cognitive effort. This higher cognitive effort could also have been influenced by the novelty of BND and participants' short exposure to it.

5.4.5 Summary

In summary, it seems the slow execution of Keyboard Shortcuts as a means to activate copy-paste commands as well as the more complex across context selection process are mostly responsible for BND's slower performance compared with DND.

5.5 How can BND be improved?

From the experiment's empirical results it is evident that the current BND implementation suffers from flaws that limit its performance. Even though mouse movement distance was lower when using BND, participants could not use this to their advantage. They ended up having slower average completion times with BND compared to DND. Hence, changes need to be made to improve BND's performance.

NASA-TLX results as well as participants' comments revealed that BND is perceived as easy to learn. It relies on the established notion of selection modes and of a clipboard. It also implements keyboard shortcuts (Ctrl-C/V) to access the clipboard. Making use of these well-known mechanisms eases the transition from traditional standard copy-paste techniques to BND. The experiment confirmed that participants, despite their brief exposure to BND, did not have any problems learning how to use it. Even though participants have had substantially more experience using KEY and DND, the qualitative results also suggested that BND was easier-to-learn than the others. In addition, participants felt it was easier to accomplish their level of performance with BND. The results confirmed that the legacy driven approach (which relies on established interaction techniques and combines them with new functionality) of the current BND makes it comparatively easy to learn (learnability).

Ironically, the experiment also showed that these well-known concepts were also responsible for BND's slower performance compared with DND. The previous section established two major flaws in BND's current implementation, namely the use of *keyboard shortcuts* and the introduction of the *across context selection mode*.

5.5.1 Selection (Bagging) mode

To improve BND, the selection process with the across context selection mode needs to be simplified. By clicking onto items while holding down 'Alt', items from different contexts can be added to the 'bag' (selection set). The across context selection mode essentially combines the previously separated selection tasks at each single context into one large, combined selection task. Beforehand, when copy-pasting from different sources, the overall copy-paste task consisted of many independent select-copy-paste subtasks that were part of the overall task. With the ability to select across contexts, these individual subtasks merged into one combined task. Therefore, when selecting items from different contexts, users needed to keep the whole selection process in working memory. Consequently, cognitively processing this larger, combined task compared to the previous smaller, independent tasks constituted a higher workload. It seems, one single, combined selection task resulted in a higher cognitive load. Also with the enlargement of the selection task, the fear of the selection mode's volatility (loss of the selection set by mistake) increased. Thus users tend to be more careful to minimise loss of selection set through mistakes, which ultimately led to a slower interaction. To counter these problems, it is necessary to break down the larger across context selection process into smaller, independent selection processes.

One possible way to break away from the concept of having one big across context selection task (in which the user selects all items from different contexts in one go) is to change the semantic meaning of the 'bag'. As it stands, when activating the across context selection mode by holding down 'Alt', selecting items at different contexts adds them to the current selection set displayed in the bag. The new selection mode has, apart from some advantages especially in terms of consistency with existing selection modes, the major disadvantage of adopting the standard selection modes' volatility. A measure to counter the volatility could be to transform the bag (with its content) to a temporary holding (storage). This storage could function like a visible clipboard that is attached to the mouse cursor. That means every item that is added to the bag is simultaneously added to this temporary storage. Hence the bag is not carrying selections (selected items) across contexts, it instead carries temporarily stored copies of the actual objects across contexts. Since the bag's items are stored and not just selected as they were before, they cannot simply be lost by accidentally clicking somewhere. As a result changing the status of the bag's items basically removes the problem of volatility. In the end, changing the meaning of the bag essentially means to discard the concept of the across context selection mode.

Another advantage of changing the bag to a temporary storage is that the overall selection process across different contexts is cut down into individual selection subtasks at each single context. Removing the volatility of the selection process, makes it harder or even impossible to lose selected/added items without explicitly confirming to. In addition, it makes it possible to intertwine the selection process with other actions, such as deleting or renaming, that would not be possible with the original BND concept. Therefore the selection and addition of items to the bag is not a continuous task anymore, rather it consists of completely separate, independent tasks at each context. Essentially, separating the across context selection task into smaller individual tasks is similar to the concept of chunking information in cognitive psychology (as described in Chapter 2.3.2.1) and therefore cuts down the user's cognitive load.

Changing the semantic meaning of the bag implicates the introduction of a new mode; the *bagging mode*. This bagging mode is instantiated quasimodal by holding down a modifier key, where items can be added to the bag via short drag-and-drop interactions. Breaking away from the concept of the across selection mode inevitably leads to the introduction of new mechanics to instantiate the bag (as a temporary storage) as well as to add new items to this bag. With the original BND concept, by holding down 'Alt' the selection mode was changed to the across context selection mode. Then continuous holding down of 'Alt' maintained this selection mode and selected items were added to the across context selection set, which was displayed in the bag. Likewise, with the improved version of BND, holding down a certain modifier key activates the new bagging mode quasimodal and displays the bag. As soon as the modifier key is released the bag disappears. In that sense the new bagging mode uses similar mechanics to instantiate the bag.

From the comments made by participants during the experiment it is questionable though, if the 'Alt' key is the most suitable choice as a modifier key to invoke this mode change. In particular the combination of Alt and Ctrl-C and the thereby associated physical shift of fingers from the Alt key to Ctrl led to a slower execution of keyboard shortcuts when using BND during the experiment. Thus the introduction of another modifier key might be more suitable. One possibility is the CapsLock key. The CapsLock key is conveniently located right above the Shift key. After having pressed the Shift and/or Ctrl key to select multiple items, the user can easily reach it using the same finger without the necessity to shift the hand (in contrast to the ‘Alt’ key). In addition, the CapsLock key’s only system-wide functionality is to invoke the caps mode. Changing into the caps mode, however, can also be done in a quasimodal way by continuously holding down Shift. This means the CapsLock key essentially does not have a unique functionality and could therefore be mapped to the activation of the bagging mode without losing any other system-wide functionality.

5.5.2 Adding items to the bag

To add items to the bag, the user drag-and-drops the selected items onto the bag. The bag, when activated, is attached to the mouse cursor and therefore follows the mouse cursor’s movement around. To be able to drag-and-drop items onto the bag, the bag exhibits a viscous movement behaviour similar to the trailing widget [Forlines, Vogel, and Balakrishnan, 2006] and the trailing lens [Ramos, Cockburn, Balakrishnan, and Beaudouin-Lafon, 2007]. The bag hereby follows the initial mouse cursor movement with a certain delay (500ms). As soon as the mouse is moved, the viscosity pulls the bag towards or pushes it away from the mouse cursor’s current position. This delay allows users to ‘catch’ the bag by quickly moving the cursor into the bag area, which thereby enables dragging and dropping items onto the bag. When the cursor leaves the bag’s area, the bag quickly aligns itself with the cursor. To add items to the bag, the user first selects the desired item(s) at the current context using either one of the different standard selection modes (Ctrl/Shift/Ctrl+Shift) or other selection mechanisms like drawing a bounding box (which were not possible to use with the previous version of BND). Then by holding down the CapsLock key, the user activates the bagging mode and the bag pops up. Since the bag moves with a delay, the user can ‘catch’ the bag by quickly dragging the selected items onto it. Once caught, the user can release the left mouse button and drop the dragged items onto the bag. This short drag-and-drop interaction is like a quick flick motion of the selected items onto the bag. As soon as the dragged items are dropped onto the bag, the user can continue selecting more items at the current context or move onto the next context add further items to the bag.

The quasimodal activation of the bagging mode combined with a flick-like drag-and-drop interaction is a simple way of adding items to the bag. Since the bag is attached to the cursor, the distance from the starting drag-point (essentially the mouse cursor’s current position) to the bag area is very short. In addition, since items only need to be dropped onto the bag and not onto a specific point of the bag, the dropping area in relation to the dragged items is large. In the experiment participants showed that they perform well when it came to drag-and-drop tasks, where the dragging distance is short and the dropping target is big and visible at all times. This short distanced drag-and-drop interaction should be similar to the drag-and-drop tasks in the experiment and therefore should constitute a fast way of adding items to the bag.

5.5.3 X copy-paste (Dumping) Pasting

To paste – or in other words ‘dump’ – the bag’s content, the user simply has to hold down the CapsLock key to display the bag (which visually reminds the user of the bag’s content) and then perform a middle mouse button click onto the destination. Since copies of the selected items are already stored in a temporary holding in the form of the bag, the items do not need to be copied to the actual system clipboard. As a result, if the system clipboard is not integrated into the workflow of BND, there is also no need to access the clipboard via keyboard shortcuts. Since the experiment confirmed that most participants (even though they are regarded as expert users) struggled with the use of keyboard shortcuts in the first place, avoiding keyboard shortcuts altogether could be an important factor in improving BND. Similar to the X Window copy-and-paste technique described in Chapter 2.4.2, the paste command is implicitly executed by determining the destination with the middle mouse button. Without the necessity of additionally pressing any keyboard shortcuts or accessing context menus with the mouse, a simple middle mouse button click onto the destination while holding down the CapsLock key determines the insertion point as well as pastes the bag's content. This drastically reduces motor operations involved to paste and thus should speed up the pasting process. Furthermore, even though this approach is similar to X copy-paste's middle mouse button pasting feature, it does not have X copy-paste's volatility since items are temporarily stored and not just selected.

An alternative pasting approach, in case the user's input device does not have a physical third button, is to directly interact with the bag using the mouse. The user hereby has to click onto the destination to determine the insertion point and then activate the bagging mode by holding down CapsLock. To dump the bag's content the user has to 'catch' the displayed bag by quickly pointing onto it. Once the bag is caught, the user can directly interact with it by clicking on a displayed button in the bag to dump the bag's content at the previously defined insertion point. In a similar manner, the user could perform other actions on the bag's content such as emptying the whole bag or selecting only certain items to paste.

5.5.4 Other future work

Further work may involve the integration of additional sensory feedback. To provide additional visual feedback for the short flicking-like drag-and-drop action, the bag could be highlighted in 'blue' as soon as the dragged items are on top of the bag to indicate the items can be dropped off. Furthermore, there could be a 'rubber band' (similar to Drag-and-Pop [Baudisch et al., 2003]) connecting the bag to the cursor. In case the cursor moves away from the bag while in bagging mode, the 'rubber band' would stretch for the time of the delay and then 'fling' the bag close to the cursor again. This would indicate that the bag is always attached to the cursor even if there is a delay in movement. Auditory feedback, similar to kinesthetic feedback, is believed to be much more effective than visual feedback because it cannot be obscured [Brewster, 1998]. Consequently, adding auditory feedback to the bagging (sound of an item falling into a bag) as well as dumping process (sound of dumping the content of a bag onto a table), could prove beneficial. While it was not a dominant issue in the experiment, future work could also investigate the best way to visualise the bag's content.

This improvement approach is basically a combination of Drag-and-Drop and X copy-paste with BND. By using short drag-and-drop interactions combined with BND's bag plus X copy-paste's middle mouse button pasting, this approach takes each techniques' advantages to overcome previous limitations. This could increase BND's performance in particular in contrast to its earlier implementation. Further experiments would be needed to evaluate this new approach.

CHAPTER 6

Conclusion

This thesis examined the problems related to copy-and-paste operations across multiple contexts and described the design and evaluation of a new copy-and-paste technique.

To fully understand the copy-and-paste paradigm, human factors that impact the capabilities and limitations during copy-and-paste operations were identified in Chapter 2. These underlying human factors were mainly related to human motor skills (pointing and dragging) as well as human cognitive information processing (memory, decision making, visual search and interruptions). Standard techniques, such as Keyboard Shortcuts, Menus and Drag-and-Drop, and existing extensions of them were then evaluated against a set of design guidelines established from the identified human factors. Even though copy-and-paste operations are regarded as one of the most fundamental services provided by today's computer systems, analysing the relative merits of these existing copy-and-paste techniques revealed obvious limitations especially in terms of efficiency and robustness against interruptions.

To overcome these limitations I proposed in Chapter 3 a new copy-and-paste technique called Bag-and-Dump. Bag-and-Dump offers the ability to 'bag' items across different folders (contexts) before 'dumping' the bagged items at the destination. A new selection mode enables users to select items across multiple contexts. In addition, visual feedback, in form of a 'bag'-like semantic cursor, displays the selected items and informs the user of the system's current state. Bag-and-Dump integrates with established concepts of selection modes and keyboard shortcuts to facilitate learnability.

Chapter 4 described an experiment that was conducted to evaluate Bag-and-Dump and compare it to two standard copy-and-paste techniques (Keyboard Shortcuts and Drag-and-Drop). All three techniques were tested in common copy-paste tasks under a different number of contexts (item mode) and with and without interruptions. The number of context varied from one, to three, to six contexts. Interruptions, in the form of mathematical problems, disrupted the workflow to test techniques' robustness against said interruptions.

Results from this experiment showed that Bag-and-Dump was 9% faster than Keyboard Shortcuts, one of the most popular copy-paste techniques. Bag-and-Dump was also the most preferred technique among participants. However, overall participants were fastest and most accurate with Drag-and-Drop, even though they had to cover a substantially longer mouse distance compared to Bag-and-Dump.

Results also revealed that participants with a higher variance in their performance across techniques were the slowest participants overall. They performed slower with Bag-and-Dump as well as with Keyboard Shortcuts mainly because they struggled to execute the copy-paste commands, which is done via keyboard shortcuts with both techniques. Another factor contributing to Bag-and-Dump's slower performance than

Drag-and-Drop was its across context selection mode, in particular the selection mode's volatility.

To counter these flaws in Bag-and-Dump's current implementation, Chapter 5 introduced possible modifications to the interaction mechanics of Bag-and-Dump to simplify the selection process as well as avoid the use of keyboard shortcuts. To remove the bag's volatility, the semantic meaning of the bag was changed from a selection set to a temporary holding of the selected items. Adding items to the bag is done with a flick motion, similar to a short drag-and-drop interaction. This avoids the use of the clipboard and therefore the need for measures to access its content, for example, via keyboard shortcuts. X copy-paste middle mouse button pasting (X-Windows system's copy-and-paste mechanism, which pastes copied clips automatically by clicking on the middle mouse button at the insertion point) was adopted to efficiently paste the bag's items at the desired destination. These modifications basically combine the advantages of Bag-and-Dump, Drag-and-Drop and X copy-paste into one single technique.

In conclusion, results from the experiment showed that Bag-and-Dump is a promising new copy-and-paste technique that outperformed at least one of the most popular techniques – Keyboard Shortcuts. Results also revealed some flaws that were discussed in more detail and modifications were proposed to address these issues. It is expected that modifications to the original interaction design of Bag-and-Dump, would lead to participants performing faster and more accurately in a future experiment.

References

Johnny Accot and Shumin Zhai. Beyond fitts' law: models for trajectory-based hci tasks. In *CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 295–302, New York, NY, USA, 1997. ACM.

Mark D. Apperley, Dale Fletcher, and William J. Rogers. Breaking the Copy/Paste Cycle: The Stretchable Selection Tool. *Australasian User Interface Conference*, 0:3, 2000.

R. C. Atkinson and R. M. Shiffrin. Human memory: A proposed system and its control processes. In K. W. Spence, editor, *The psychology of learning and motivation: Advances in research and theory*, pages 89–195. Academic Press, New York, 1968.

Christopher Baber. *Beyond the Desktop: Designing and Using Interaction Devices (Computers and People)*. Academic Press, 1996.

A. D. Baddeley. *Working Memory*. Oxford University Press, New York, 1986.

A. D. Baddeley. Working memory. *Science*, 255:556–559, 1992.

A. D. Baddeley and G. J. Hitch. Working memory. In G. Bower, editor, *The psychology of learning and motivation*, volume VIII, pages 47–89. Academic Press, New York, 1974.

Alan Baddeley. *Human Memory: Theory and Practice, Revised Edition*. Psychology Press, 1997.

Alan Baddeley. The episodic buffer: a new component of working memory? *Trends in Cognitive Sciences*, 4(11):417 – 423, 2000.

Alan D. Baddeley, Neil Thomson, and Mary Buchanan. Word length and the structure of short-term memory. *Journal of Verbal Learning and Verbal Behavior*, 14(6):575 – 589, 1975.

P. Baudisch, E. Cutrell, D. Robbins, and M. Czerwinski. Drag-and-Pop and Drag-and-Pick: Techniques for Accessing Remote Screen Content on Touch- and Pen-Operated Systems. In *INTERACT*, 2003.

Eric A. Bier, Edward W. Ishak, and Ed Chi. Entity quick click: rapid text copying based on automatic entity extraction. In *CHI '06: CHI '06 extended abstracts on Human factors in computing systems*, pages 562–567, New York, NY, USA, 2006. ACM. doi: http://doi.acm.org/10.1145/1125451.1125570.

Renaud Blanch, Yves Guiard, and Michel Beaudouin-Lafon. Semantic pointing: improving target acquisition with control-display ratio adaptation. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 519–526, New York, NY, USA, 2004. ACM. doi: http://doi.acm.org/10.1145/985692.985758.

Florian Block, Nicolas Villar, and Hans Gellersen. A malleable physical interface for copying, pasting, and organizing digital clips. In *TEI '08: Proceedings of the 2nd international conference on Tangible and embedded interaction*, pages 117–120, New York, NY, USA, 2008. ACM. doi: http://doi.acm.org/10.1145/1347390.1347415.

Stephen A Brewster. Sonically-enhanced drag and drop. In *BRITISH COMPUTER SOCIETY*, 1998.

Stephen A. Brewster, Peter C. Wright, and Alistair D. N. Edwards. An evaluation of earcons for use in auditory human-computer interfaces. In *CHI '93: Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*, pages 222–227, New York, NY, USA, 1993. ACM. doi: http://doi.acm.org/10.1145/169059.169179.

Stuart K. Card, Thomas P. Moran, and Allen Newell. The keystroke-level model for user performance time with interactive systems. *Commun. ACM*, 23(7):396–410, 1980. doi: http://doi.acm.org/10.1145/358886.358895.

Géry Casiez, Daniel Vogel, Qing Pan, and Christophe Chaillou. Rubberedge: reducing clutching by combining position and rate control with elastic feedback. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*, UIST '07, pages 129–138, New York, NY, USA, 2007. ACM. doi: http://doi.acm.org/10.1145/1294211.1294234.

Jean-Marie Cellier and Hélène Eyrolle. Interference between switched tasks. *Ergonomics*, 35:25 – 36, 1992.

Olivier Chapuis and Nicolas Roussel. Copy-and-paste between overlapping windows. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 201–210, New York, NY, USA, 2007. ACM. doi: http://doi.acm.org/10.1145/1240624.1240657.

Olivier Chapuis, Jean-Baptiste Labrune, and Emmanuel Pietriga. Dynaspot: speed-dependent area cursor. In *CHI '09: Proceedings of the 27th international conference* *on Human factors in computing systems*, pages 1391–1400, New York, NY, USA, 2009. ACM. doi: http://doi.acm.org/10.1145/1518701.1518911.

Wayne Citrin, Daniel Broodsky, and Jeffrey McWhirter. Style-based cut-and-paste in graphical editors. In *AVI '94: Proceedings of the workshop on Advanced visual interfaces*, pages 105–112, New York, NY, USA, 1994. ACM. doi: http://doi.acm.org/10.1145/192309.192331.

J. Cohen. *Applied multiple regression/correlation analysis for the behavioral sciences*. Lawrence Erlbaum, 2003.

Maxime Collomb, Mountaz Hascoët, Patrick Baudisch, and Brian Lee. Improving drag-and-drop on wall-size displays. In *GI '05: Proceedings of Graphics Interface 2005*, pages 25–32, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2005. Canadian Human-Computer Communications Society.

H. S. M. Coxeter. *Introduction to Geometry (Wiley Classics Library)*. Wiley, 1989.

Edward B. Cutrell, Mary Czerwinski, and Eric Horvitz. Effects of instant messaging interruptions on computing tasks. In *CHI '00: CHI '00 extended abstracts on Human factors in computing systems*, pages 99–100, New York, NY, USA, 2000. ACM. doi: http://doi.acm.org/10.1145/633292.633351.

Mary Czerwinski, Eric Horvitz, and Susan Wilhite. A diary study of task switching and interruptions. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 175–182, New York, NY, USA, 2004. ACM. doi: http://doi.acm.org/10.1145/985692.985715.

Alan Dix, Janet Finlay, Gregory D. Abowd, and Russell Beale. *Human Computer Interaction*. Pearson, Harlow, England, 3. edition, 2003.

Pierre Dragicevic. Combining crossing-based and paper-based interaction paradigms for dragging and dropping between overlapping windows. In *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 193–196, New York, NY, USA, 2004. ACM. doi: http://doi.acm.org/10.1145/1029632.1029667.

Elizabeth Dykstra-Erickson and Dave Curbow. The role of user studies in the design of opendoc. In *DIS '97: Proceedings of the 2nd conference on Designing interactive systems*, pages 111–120, New York, NY, USA, 1997. ACM. doi: http://doi.acm.org/10.1145/263552.263588.

Guillaume Faure, Olivier Chapuis, and Nicolas Roussel. Power tools for copying and moving: useful stuff for your desktop. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 1675–1678, New York, NY, USA, 2009. ACM. doi: http://doi.acm.org/10.1145/1518701.1518958.

Paul M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47(6):381 – 391, 1954.

George Fitzmaurice, Justin Matejka, Azam Khan, Michael Glueck, and Gordon Kurtenbach. Piecursor: merging pointing and command selection for rapid in-place tool switching. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1361–1370, New York, NY, USA, 2008. ACM. doi: http://doi.acm.org/10.1145/1357054.1357268.

Clifton Forlines, Daniel Vogel, and Ravin Balakrishnan. Hybridpointing: fluid switching between absolute and relative pointing with a direct input device. In *UIST '06: Proceedings of the 19th annual ACM symposium on User interface software* *and technology*, pages 211–220, New York, NY, USA, 2006. ACM. doi: http://doi.acm.org/10.1145/1166253.1166286.

William W. Gayer. The sonicfinder: An interface that uses auditory icons (abstract only). *SIGCHI Bull.*, 21(1):124, 1989. doi: http://doi.acm.org/10.1145/67880.1046601.

Jörg Geissler. Shuffle, throw or take it! working efficiently with an interactive wall. In *CHI '98: CHI 98 conference summary on Human factors in computing systems*, pages 265–266, New York, NY, USA, 1998. ACM. doi: http://doi.acm.org/10.1145/286498.286745.

Tovi Grossman and Ravin Balakrishnan. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 281–290, New York, NY, USA, 2005. ACM. doi: http://doi.acm.org/10.1145/1054972.1055012.

Y. Guiard, F. Bourgeois, D. Mottet, and M. Beaudouin-Lafon. Beyond the 10-bit barrier: Fitts' law in multi-scale electronic worlds. In *IHM-HCI 2001*, 2001.

Rikard Harr and Victor Kaptelinin. Unpacking the social dimension of external interruptions. In *GROUP '07: Proceedings of the 2007 international ACM conference on Supporting group work*, pages 399–408, New York, NY, USA, 2007. ACM. doi: http://doi.acm.org/10.1145/1316624.1316686.

S.G. Hart and L.E. Staveland. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. *Human mental workload*, 1:139–183, 1988.

M. Hascoet. Throwing models for large displays. *In Proc. HCI'03*, 2:73–77, 2003.

W. E. Hick. On the rate of gain of information. *The Quarterly Journal of Experimental Psychology*, 4:11 – 26, 1952.

Ken Hinckley, Patrick Baudisch, Gonzalo Ramos, and Francois Guimbretiere. Design and analysis of delimiters for selection-action pen gesture phrases in scriboli. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 451–460, New York, NY, USA, 2005. ACM. doi: http://doi.acm.org/10.1145/1054972.1055035.

Ken Hinckley, Francois Guimbretiere, Patrick Baudisch, Raman Sarin, Maneesh Agrawala, and Ed Cutrell. The springboard: multiple modes in one spring-loaded control. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 181–190, New York, NY, USA, 2006. ACM. doi: http://doi.acm.org/10.1145/1124772.1124801.

Dugald Ralph Hutchings, Greg Smith, Brian Meyers, Mary Czerwinski, and George Robertson. Display space usage and window management operation comparisons between single monitor and multiple monitor users. In *Proceedings of the working conference on Advanced visual interfaces*, AVI '04, pages 32–39, New York, NY, USA, 2004. ACM. doi: http://doi.acm.org/10.1145/989863.989867.

Ray Hyman. Stimulus information as a determinant of reaction time. *Journal of Experimental Psychology*, 45(3):188 – 196, 1953.

Patricia Jablonski. Managing the copy-and-paste programming practice in modern ides. In *OOPSLA '07: Companion to the 22nd ACM SIGPLAN conference on Object-oriented programming systems and applications companion*, pages 933–934, New York, NY, USA, 2007. ACM. doi: http://doi.acm.org/10.1145/1297846.1297952.

Robert L. Mack Jakob Nielsen. *Usability Inspection Methods*. Wiley, 1994.

Jackie Andrade Janet D. Larsen, Alan Baddeley. Phonological similarity and the irrelevant speech effect: Implications for models of short-term verbal memory. *Memory*, 8 (3):145–157, 2000.

Herbert D. Jellinek and Stuart K. Card. Powermice and user performance. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people*, CHI '90, pages 213–220, New York, NY, USA, 1990. ACM. doi: http://doi.acm.org/10.1145/97243.97276.

Jing Jin and Laura A. Dabbish. Self-interruption on the computer: a typology of discretionary task interleaving. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 1799–1808, New York, NY, USA, 2009. ACM. doi: http://doi.acm.org/10.1145/1518701.1518979.

Jeff Johnson, Teresa L. Roberts, William Verplank, David C. Smith, Charles H. Irby, Marian Beard, and Kevin Mackey. The xerox star: A retrospective. *Computer*, 22(9): 11–26, 28–29, 1989. doi: http://dx.doi.org/10.1109/2.35211.

Reid Kerr and Wolfgang Stuerzlinger. Context-sensitive cut, copy, and paste. In *C3S2E '08: Proceedings of the 2008 C3S2E conference*, pages 159–166, New York, NY, USA, 2008. ACM. doi: http://doi.acm.org/10.1145/1370256.1370283.

Miryung Kim, Lawrence Bergman, Tessa Lau, and David Notkin. An ethnographic study of copy and paste programming practices in oopl. In *ISESE '04: Proceedings of the 2004 International Symposium on Empirical Software Engineering*, pages 83–92, Washington, DC, USA, 2004. IEEE Computer Society. doi: http://dx.doi.org/10.1109/ISESE.2004.10.

Masatomo Kobayashi and Takeo Igarashi. Boomerang: suspendable drag-and-drop interactions based on a throw-and-catch metaphor. In *UIST '07: Proceedings of the*

20th annual ACM symposium on User interface software and technology, pages 187–190, New York, NY, USA, 2007. ACM. doi: http://doi.acm.org/10.1145/1294211.1294243.

Melina A. Kunar, Stephen Flusberg, and Jeremy M. Wolfe. The role of memory and restricted context in repeated visual search. *Perception & Psychophysics*, 70(2):314 – 328, 2008.

D. Lane, H. Napier, S. Peres, and A. Sandor. Hidden costs of graphical user interfaces: Failure to make the transition from menus and icon toolbars to keyboard shortcuts. In *International Journal of Human-Computer Interaction*, volume 18, pages 133–144, 2005.

I. Scott MacKenzie. Fitts' law as a research and design tool in human-computer interaction. *Hum.-Comput. Interact.*, 7(1):91–139, 1992. doi: http://dx.doi.org/10.1207/s15327051hci0701_3.

I. Scott MacKenzie and Aleks Oniszczak. A comparison of three selection techniques for touchpads. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '98, pages 336–343, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co. doi: http://dx.doi.org/10.1145/274644.274691.

I. Scott Mackenzie and Stan Riddersma. Effects of output display and control-display gain on human performance in interactive systems. *Behaviour & Information Technology*, 13:328 – 337, 1994.

I. Scott MacKenzie, Abigail Sellen, and William A. S. Buxton. A comparison of input devices in element pointing and dragging tasks. In *CHI '91: Proceedings of the*

SIGCHI conference on Human factors in computing systems, pages 161–166, New York, NY, USA, 1991. ACM. doi: http://doi.acm.org/10.1145/108844.108868.

Ian Scott Mackenzie. *Fitts' law as a performance model in human-computer interaction*. PhD thesis, University of Toronto, Toronto, Ont., Canada, Canada, 1991.

Regan L. Mandryk and Carl Gutwin. Perceptibility and utility of sticky targets. In *GI '08: Proceedings of graphics interface 2008*, pages 65–72, Toronto, Ont., Canada, Canada, 2008. Canadian Information Processing Society.

Gloria Mark, Victor M. Gonzalez, and Justin Harris. No task left behind?: examining the nature of fragmented work. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 321–330, New York, NY, USA, 2005. ACM. doi: http://doi.acm.org/10.1145/1054972.1055017.

Margaret W. Matlin. *Cognition*. Wiley, 2008.

Michael McGuffin and Ravin Balakrishnan. Acquisition of expanding targets. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 57–64, New York, NY, USA, 2002. ACM. doi: http://doi.acm.org/10.1145/503376.503388.

John Medina. *Brain Rules: 12 Principles for Surviving and Thriving at Work, Home, and School*. Pear Press, 2009.

George Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information, 1956.

Jakob Nielsen. *Usability Engineering*. Morgan Kaufmann Publishers, San Francisco, California, October 1994.

Takeshi Nishida and Takeo Igarashi. Drag-and-guess: Drag-and-drop with prediction. In *INTERACT*, pages 461–474, 2007.

D.A. Norman. Categorization of action slips. *Psychological review*, 88(1):1–15, 1981.

D.A. Norman. *The design of everyday things*, volume 16. Basic Books New York, 2002.

Brid O'Conaill and David Frohlich. Timespace in the workplace: dealing with interruptions. In *CHI '95: Conference companion on Human factors in computing systems*, pages 262–263, New York, NY, USA, 1995. ACM. doi: http://doi.acm.org/10.1145/223355.223665.

S. C. Peres, M. D. Fleetwood nad F. P. Tamborello II, M. Yang, and D. L. Paige-Smith. Pros, cons, and changing behavior: An application in using the keyboard to issue commands. In *Proceedings of Human Factors and Ergonomics Society 49th Annual Meeting*, pages 637–641, 2005.

Matthew S. Peterson, Arthur F. Kramer, Ranxiao Frances Wang, David E. Irwin, and Jason S. McCarley. Visual search has memory. *Psychological Science*, 12:287–292(6), 2001.

M.F. Poller and S.K. Garter. The effects of modes on text editing by experienced editor users. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 26(4):449–462, 1984.

Gonzalo Ramos, Andy Cockburn, Ravin Balakrishnan, and Michel Beaudouin-Lafon. Pointing lenses: facilitating stylus input through visual-and motor-space magnification. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 757–766, New York, NY, USA, 2007. ACM. doi: http://doi.acm.org/10.1145/1240624.1240741.

Jef Raskin. *The Humane Interface: New Directions for Designing Interactive Systems*. Addison-Wesley Professional, 2000.

Jun Rekimoto. Pick-and-drop: a direct manipulation technique for multiple computer environments. In *UIST '97: Proceedings of the 10th annual ACM symposium on User interface software and technology*, pages 31–39, New York, NY, USA, 1997. ACM. doi: http://doi.acm.org/10.1145/263407.263505.

L. Renard. Cut and paste 101: Plagiarism and the net. *Educational Leadership*, 57(4): 38–42, 1999.

R.D. Rogers and S. Monsell. Costs of a predictable switch between simple cognitive tasks. *Journal of Experimental Psychology-General*, 124(2):207–230, 1995.

J. Rubinstein, D. Meyer, and J. Evans. Executive control of cognitive processes in task switching. *Journal of Experimental Psychology*, 27(4):763–797, 2001.

Colin Runciman and Harold Thimbleby. Equal opportunity interactive systems. *International Journal of Man-Machine Studies*, 25(4):439 – 451, 1986.

Dewey Rundus. Analysis of rehearsal processes in free recall. *Journal of Experimental Psychology*, 89(1):63 – 77, 1971.

Mark Sanders and Ernest McCormick. *Human Factors In Engineering and Design*. McGraw-Hill Science/Engineering/Math, 1993a.

Mark Sanders and Ernest McCormick. *Human Factors In Engineering and Design*. McGraw-Hill Science/Engineering/Math, 1993b.

R.W. Scheifler and J. Gettys. The X window system. *ACM Transactions on Graphics (TOG)*, 5(2):79–109, 1986.

A.J. Sellen, G.P. Kurtenbach, and W.A.S. Buxton. The prevention of mode errors through sensory feedback. *Human-Computer Interaction*, 7(2):141–164, 1992.

Claude E. Shannon and Warren Weaver. *A Mathematical Theory of Communication*. University of Illinois Press, Champaign, IL, USA, 1963.

Dr Helen Sharp, Professor Yvonne Rogers, and Dr Jenny Preece. *Interaction Design: Beyond Human-Computer Interaction*. Wiley, 2007.

Ben Shneiderman. Creating creativity: user interfaces for supporting innovation. *ACM Trans. Comput.-Hum. Interact.*, 7(1):114–138, 2000. doi: http://doi.acm.org/10.1145/344949.345077.

G. Smith, P. Baudisch, G. Robertson, M. Czerwinski, B. Meyers, D. Robbins, and D. Andrews. Groupbar: The taskbar evolved. In *Proc. OzCHI*, volume 3. Citeseer, 2003.

R. William Soukoreff and I. Scott MacKenzie. Towards a standard for pointing device evaluation, perspectives on 27 years of fitts' law research in hci. *Int. J. Hum.-Comput. Stud.*, 61(6):751–789, 2004. doi: http://dx.doi.org/10.1016/j.ijhcs.2004.09.001.

Jeffrey Stylos, Brad A. Myers, and Andrew Faulring. Citrine: providing intelligent copy-and-paste. In *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 185–188, New York, NY, USA, 2004. ACM. doi: http://doi.acm.org/10.1145/1029632.1029665.

Ivan E. Sutherland. Sketchpad: a man-machine graphical communication system. In *AFIPS '63 (Spring): Proceedings of the May 21-23, 1963, spring joint computer conference*, pages 329–346, New York, NY, USA, 1963. ACM. doi: http://doi.acm.org/10.1145/1461551.1461591.

Susanne Tak. The use of keyboard shortcuts; optimizing versus satisficing in the use of complex technology. Master's thesis, Eindhoven University of Technology, August 2007.

Susanne Tak and Andy Cockburn. Improved window switching interfaces. In *CHI EA '10: Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems*, pages 2915–2918, New York, NY, USA, 2010. ACM. doi: http://doi.acm.org/10.1145/1753846.1753884.

G. Treisman, A.; Gelade. A feature integration theory of attention. *Cognitive Psychology*, 12:97 – 136, 1980.

Annette Wagner, Patrick Curran, and Robert O'Brien. Drag me, drop me, treat me like an object. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 525–530, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co. doi: http://doi.acm.org/10.1145/223904.223975.

Glen Wallace, Robert Biddle, and Ewan Tempero. Smarter cut-and-paste for programming text editors. In *AUIC '01: Proceedings of the 2nd Australasian conference on User interface*, pages 56–63, Washington, DC, USA, 2001. IEEE Computer Society.

C. Wallis. The multitasking generation. *Time Magazine*, 167(13):48–56, 2006.

J.M. Wolfe, T.S. Horowitz, and N.M. Kenner. Rare items often missed in visual searches. *Nature*, 435(7041):439–440, 2005.

Miyata Y. and Norman D. Psychological issues in support of multiple activities. *User Centered Systems Design: New Perspectives on Human-Computer Interaction*, pages 265–284, 1986.