

Revealing the Copy and Paste Habits of End Users

Kathryn T. Stolee, Sebastian Elbaum, and Gregg Rothermel
Department of Computer Science and Engineering
University of Nebraska – Lincoln
Lincoln, NE, U.S.A.
{kstolee, elbaum, grother}@cse.unl.edu

Abstract

Transferring data across applications is a common end user task, and copying and pasting via the clipboard lets users do so relatively easily. Using the clipboard, however, can also introduce inefficiencies and errors in user tasks. To help researchers and tool developers understand and address these problems, we studied how end users interact with the clipboard through cut, copy, and paste actions. This study was performed by logging clipboard interactions while end users performed everyday tasks. From the clipboard usage data, we have identified several usage patterns that describe how data is transferred within the desktop environment. Such patterns help us understand end user behavior and indicate areas in which clipboard support tools can be improved.

1. Introduction

End users who wish to move data often do so by copying it onto a clipboard and later pasting it into a chosen destination. Consider for example Alex, an administrative assistant and one of the millions of end users who are programming today [9]. Every month Alex must compile an expense report for his division. To do this, he retrieves last month's report and repeatedly copies data from emails, internal business system websites, and spreadsheets onto the clipboard – all of these are common activities performed by administrative assistants [10]. Alex then pastes that data into the report's fields. By using the clipboard, Alex saves time and avoids data transcription errors that could have occurred through manually re-entering the data.

Despite the convenience and simplicity of copying and pasting through the clipboard and its usefulness in reducing transcription errors, clipboard usage can also introduce other forms of errors. Among professional programmers, copy and paste activities have been found to introduce errors via pasting of incomplete copies and pasting without proper context updates. These errors were seen to reduce productivity by delaying development during program maintenance

tasks [6]. Context switches caused by copy and paste activities across overlapping windows may also reduce productivity by causing delays in task completion [2]. There have been some efforts to explore clipboard usage and the purpose of copy and paste actions by professional programmers working in controlled environments, such as within IDEs [5, 13]. Also, several tools have been developed to support and to address inefficiencies in copy and paste activities [1, 2, 3, 7, 12].

In spite of this work, we do not yet know much about how *end users* use the clipboard, to what extent they use it, where it is used most often, or what copy and paste patterns represent their behavior. Understanding these issues is important as it directly affects end users' productivity and the correctness of the tasks they perform, with implications for the environments and tools that might support those tasks.

In this paper we address these issues by studying the copy and paste behaviors of end users, and based on the data obtained, defining patterns that represent end user behaviors. We discuss ways in which these patterns, in conjunction with our other findings regarding clipboard usage, have implications for researchers and tool builders who wish to support automation of data transfer tasks or enhance validation of copy-paste activities to increase dependability and productivity.

The next section of this paper describes our approach for studying clipboard usage and the data we collected. Section 3 presents the patterns we inferred from the study data and Section 4 describes the implications of these patterns and the usage data for further research and support for clipboard activities. Section 5 summarizes our findings.

2. Clipboard Study

To explore how end users utilize the clipboard, we built a tool that unobtrusively monitors a user's clipboard activities and remotely logs the collected observations. Then, we conducted a study on a group of end users and analyzed the collected data to determine how often the clipboard was used, with what kinds of applications it was used, and which

applications were typically used together with the clipboard as a means for data transfer. We now describe these steps in more detail.

2.1. Setup and Methodology

Our clipboard monitor keeps track of the type of clipboard interaction (i.e., cut, copy, or paste), which method was used for the interaction (i.e., keyboard or mouse), time and date, the application involved, window title, data value, and data size. The monitor anonymizes any data that can identify a user to minimize privacy concerns and potential changes in user behavior. This is done by performing a hash on the copied data and on window titles associated with the applications being used, which allows us to recognize when the same window was used again as well as when the same textual data was copied more than once. To better fit the target user population, the monitor was built to work with the Windows XP and Vista clipboards.

We distributed the monitor to two groups of end users: the first was composed of administrative assistants and teachers, and the second was composed of students who are non-CS-majors in an introductory computer science course. Our sample selection criteria intended to capture two distinct end user groups that are readily available within our institution. The participants were asked to install the monitoring software and leave it running on their computers for four weeks while performing their regular activities. In total, 25 users downloaded our tool and 15 followed through with the study. Seven of those users came from the first group and eight came from the second group. Our average observation time per user was over 50 hours.

A total of 2544 clipboard interactions were captured; 1158 (45%) were copies, 71 (3%) were cuts, and 1315 (52%) were pastes. The average number of clipboard interactions per hour per user was between three and four – considerably smaller than the 16 copy-paste instances per hour reported for professional programmers [5]. Note, however, that our study observed end users during all computer activities – not just those related to programming – which may account for these differences. We also observed some variability in usage frequency among end users, with one user’s average being nearly 20 clipboard interactions per hour.

2.2. Findings

We treat cuts and copies the same, as both perform the same action from the perspective of the clipboard – both place data onto the clipboard – and the number of cuts performed was small compared to the number of copies and pastes. For each item that was copied to the clipboard, we looked at the number of destinations for that item, which is the number of locations in which a paste of that item occurred. We found that data put onto the clipboard was rarely

Table 1. Application Category Usage

Application Category	% of Use	% as Source – Destination
Word Processors	26%	36% – 64%
Web Browsers	23%	58% – 42%
Email Clients	19%	49% – 51%
Spreadsheets	18%	51% – 49%
IDEs	5%	40% – 60%
File System	4%	44% – 56%
Business	3%	61% – 39%
Other	2%	81% – 19%

wasted – approximately 81% of copied data was pasted. Regarding the wasted data, our intuition is that it represents cases when a user accidentally copies data or copies the wrong data and realizes it before pasting. We also found that pasted data had a single destination 83% of the time. Of those data items that were pasted multiple times, the average number of destinations was approximately three.

We also examined how end users treated particular categories of applications as well as how often each category was used as a data source (i.e., in a copy), or a destination (i.e., in a paste). We considered eight different categories, shown in Table 1. The *% of Use* column shows the percentage of all clipboard interactions that involved the category, and the *% as Source – Destination* column shows how often the category was used as a source or as a destination. Overall, word processors were the most popular type of application and were used predominantly as a destination. On the other hand, web browsers were used more as a source than a destination. Spreadsheets and email clients served equally ($\pm 2\%$) as sources and destinations. The last four categories exhibited comparatively little usage.

Each user interacted with between two and 16 distinct applications. The frequency of application use per user, or how often each application was used with the clipboard, showed that users tended to concentrate their copy and paste actions within a small number of applications. On average, users performed nearly 80% of their clipboard interactions using two distinct applications, 88% using three applications, and nearly 97% within five applications. While the actual applications used varied per user, the trend of concentrating clipboard activities in just a few applications was consistent.

To gauge the interactions between specific applications, we examined the most popular applications used with the clipboard across all users, shown in Figure 1. Each application is represented as a node, and there is an additional node representing the 20 applications with the least usage that are not explicitly shown. The solid edges represent links created by a copy and one of its subsequent pastes; these are undirected to show simply that a link exists between two applications. The dashed loops indicate self-loops in which

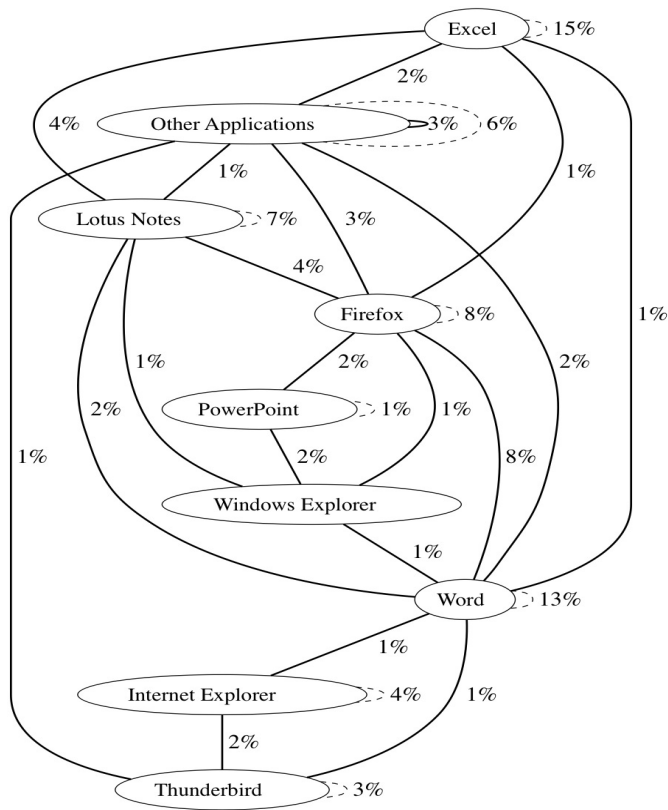


Figure 1. Interaction percentages among popular applications across all users

the same application served as both source and destination. The edge labels represent a percentage of the total copy and paste interactions that occurred between applications (e.g., 4% of the total links created by a copy and a subsequent paste occurred between Firefox and Lotus Notes).

Figure 1 reveals interesting differences among copy and paste activities across applications. For some applications like Excel, most copies and pastes occurred within the application, while for others like PowerPoint, most copies and pastes transferred data across applications. Overall, we found that approximately 43% of pasted data had been copied from within a different application.

3. Discovering Behavior Patterns

We now proceed to analyze the collected data to uncover patterns in the users behavior. For that, we perform a finer level of analysis and consider user copy and paste behavior between *windows*, where an application may have multiple windows (e.g., worksheets in spreadsheets, tabs in browsers, the find dialog for a word processor). To help us present the patterns we discovered in clipboard usage at the window level, we define a *transaction pair* as one copy followed by one paste, which represents a user’s explicit movement of data from one window to another. This con-

Table 2. Elementary Patterns

Description	Graph	Usage	
		Per user	Overall
Between. A pattern involving one transaction pair in which the source window is not the same as the destination window.		$\mu: 77\%$ $\sigma: 18\%$	65%
Within. A pattern involving one transaction pair in which the source and destination window are the same.		$\mu: 23\%$ $\sigma: 18\%$	35%

cept fits naturally with a copy that is followed by only one paste. For copies that are pasted into multiple destinations, as when a copy in window x is followed by a paste into window y and then a paste into window z , we treat it as two sequential transaction pairs; the same data item is moved via the clipboard from x to y and also from x to z .

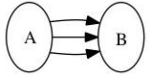
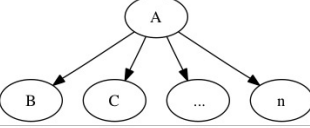
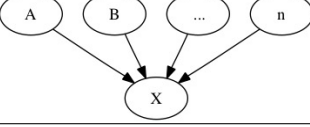
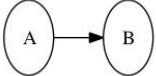
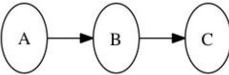
In the next sections, we describe two classes of patterns, elementary and complex. A *pattern instance* is an occurrence of a pattern in the data. For the elementary patterns, each pattern instance is composed of exactly one transaction pair. For the complex patterns, each pattern instance is composed of one or more transaction pairs. The frequency with which these patterns were observed helps us to understand, describe, and make inferences regarding user behaviors.

3.1. Elementary Patterns

Through studying the user data, we observed and defined two elementary patterns that represent the relationship between the source window and the destination window in any transaction pair. These patterns are shown in Table 2. The *Description* column gives the pattern name and a short textual description. The *Graph* column visually depicts the pattern, where the nodes represent windows and the arrows represent the directional movement of data. The *Usage* columns indicate the frequency with which transaction pairs fit into a particular pattern. The *Per user Usage* column considers the pattern frequency per user and presents the average and standard deviation. The *Overall Usage* column considers the user population as a whole, and presents the pattern frequency for all users.

The *between* pattern involves a single transaction pair in which the source window differs from the destination window. For this pattern to occur, the user must perform a change in window focus (navigate away from the currently active window to another window) and may also perform a context switch (bring to focus part or all of a window that was previously hidden from view) between the source window and destination window. The *within* pattern involves

Table 3. Complex Patterns

Description	Graph	Usage	
		Per user	Overall
Repeat. A pattern involving two or more consecutive transaction pairs in which every pair’s source window is the same and every pair’s destination window is the same, and the source window is not the same as the destination window.		μ : 32% σ : 18%	37%
Distribution. A pattern involving two or more consecutive transaction pairs in which one common source window feeds data into two or more distinct destination windows.		μ : 36% σ : 16%	32%
Composition. A pattern involving two or more consecutive transaction pairs in which two or more distinct source windows feed data into one common destination window.		μ : 19% σ : 17%	18%
Isolation. A pattern involving one transaction pair in which neither the source window nor the destination window is the same as the windows in the transaction pairs occurring immediately before or immediately after it.		μ : 10% σ : 8%	10%
Relay. A pattern that is composed of two consecutive transaction pairs in which the first transaction’s destination window is the same as the second transaction’s source window, and all the windows are different.		μ : 2% σ : 2%	3%

a single transaction pair in which the source window is the same as the destination window. The user does not need to switch windows in order to perform the paste, thus no changes in window focus are necessary in this case.

In our study, we observed that data was transferred between windows more often than within the same window: 65% of all transaction pairs fit the between pattern and individual users exhibited an average between pattern frequency of 77%. This is compared to the within pattern frequency of 35% overall and 23% per user. The difference in frequencies indicates that the users for whom we logged more clipboard interactions showed lower frequencies of the between pattern than the other users did.

3.2. Complex Patterns

By considering relationships between instances of the elementary patterns, we have identified several complex patterns that involve aggregations of the elementary patterns. These complex patterns represent temporal relationships among the transaction pairs. The high frequency of the between pattern and our interest in data movement across windows led us to focus only on complex patterns that involve instances of the between pattern. We present these complex patterns in Table 3. As with Table 2, the *Usage* column considers the frequency with which transaction pairs occur in the specific patterns, considering only the transaction pairs involved with the patterns presented here. Since the average

pattern frequency per user is similar ($\pm 5\%$) to the overall frequency for all patterns, we treat the user population as a whole for this analysis. Note that every instance of the between pattern fits at least one of these complex patterns, except for the rare case of two sequential transaction pairs in which the source of the first is the same as the destination of the second; this pattern did not imply intuitive user behavior and had the lowest support of all the patterns (less than 2%), so we omitted it from the analysis and normalized the usage frequencies.

Over a third (37%) of the transaction pairs appeared in a repeat pattern instance. This pattern involves multiple sequential transaction pairs in which the source window in all pairs is the same and the destination window in all pairs is also the same, yet the source window and destination windows are distinct. The average number of transaction pairs per repeat pattern was slightly over three. The high percentage of occurrences of this pattern indicates that users often worked repetitively between two windows and performed copies and pastes in a directional manner from one window to another. This pattern requires multiple changes in window focus and may also involve rapid context switches.

Close in frequency to the repeat pattern is the distribution pattern, which describes 32% of the transaction pairs. This pattern represents the case of a user working within one window and distributing data to two or more different windows, such as copying an address from a website into a

spreadsheet and then an address book. The average number of pastes per distribution pattern instance was slightly over four. This pattern does not require as many changes in window focus as the repeat pattern: one copy can be pasted into multiple destinations without re-visiting the source window.

Following the distribution pattern in frequency is the composition pattern, which represents 18% of all transaction pairs. This pattern is the reverse of the distribution pattern and involves the aggregation of data from two or more distinct source windows into one common destination window. The average number of copies per composition pattern instance was close to five. Alex, the administrative assistant from our example, would likely use a composition pattern to gather data from multiple sources and compile them into a single destination.

The next pattern is the isolation pattern which is seen in 10% of the transaction pairs. This pattern involves one transaction pair with distinct source and destination windows (i.e., not an instance of the within pattern), and requires that neither the source window nor the destination window is the same as either of the windows within the transaction pairs occurring immediately before or after it. What is most interesting about this pattern is its low frequency, implying that 90% of the transaction pairs had some direct relationship with the pair coming immediately before or immediately after.

Finally, the relay pattern accounts for 3% of all transaction pairs. This pattern represents two sequential transaction pairs in which the destination of the first pair is the same as the source of the second. An instance of this pattern may occur when a user pastes data into a temporary destination for reformatting and copies it again onto the clipboard before pasting into its final destination.

4. Discussion

Ours is just one study of end user clipboard usage and further studies are clearly necessary to generalize our results. Still, end user behavior seems to follow a set of patterns representing their treatment of source and destination windows. These patterns, in conjunction with our other findings regarding clipboard usage, have implications for researchers and tool builders who wish to support automation of data transfer tasks or validation of copy-paste activities to increase dependability and improve user productivity.

For example, end users often utilize the clipboard to transfer data within and between windows, but transfers between windows happen much more frequently. End users also perform the majority of their clipboard interactions within a handful of applications, though the specific applications used vary based on the user. From these observations, we can infer the need for a clipboard support tool that *works with all desktop applications*. This feature, as well as the other features we identify in this section, appear in

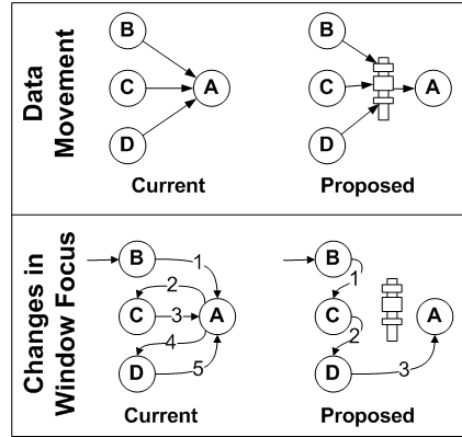


Figure 2. Multiple-item clipboard, shown with composition pattern

Table 4, along with some state-of-the-art clipboard support tools, each of which is compared against the 12 support features we identify. Note that we are not advocating specific tools to support these user behaviors, but rather possible *features* that future tools could provide to better fit user needs. These support needs are described in the following sections.

4.1. Automation

There are several ways in which automation could be applied to a clipboard support tool. Here, we identify several possible automation features.

4.1.1 Multiple-Item Clipboard

The high frequency of the repeat pattern shows that users often switch between windows repeatedly to perform related copy and paste actions. Such extra work could be alleviated by a clipboard that *holds multiple items at once*. For example, if a user copies an address to paste into a form, a multiple-item clipboard could hold each of the street address, city, state, and zip code as a separate item, allowing the data to be accessed in the destination application without returning to the source.

Such a concept would also be beneficial to users when they employ the composition pattern, in which items are copied from multiple sources and pasted into a single destination. The intuition behind the composition pattern is that users perform actions that are centered around one particular window in which they are operating. This window serves as the destination for copies that are aggregated from multiple sources. In the composition pattern, each copy requires a change in focus to the destination window for a paste, then a return to another source window for a copy. A multiple-item clipboard would reduce the number of times the user would return to the destination window from each source.

We illustrate the concept of the multiple-item clipboard in Figure 2 using an instance of the composition pattern

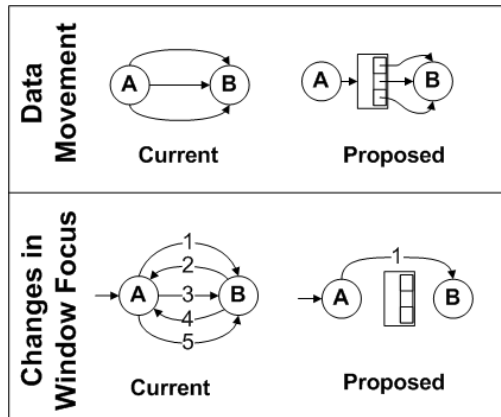


Figure 3. Context aware clipboard, shown with repeat pattern

as an example. The *Current* images show the traditional clipboard model, while the *Proposed* images show how the multiple-item clipboard would assist with the pattern. The top box illustrates the data movement from windows *B*, *C*, and *D* into *A*; the bottom box illustrates the changes in window focus that are necessary given the data movement. The key to this solution is that copying data onto and pasting data from the multiple-item clipboard does not involve changes in window focus. With the composition pattern, we see a clear reduction in the number of changes in window focus using a multiple-item clipboard. The multiple-item clipboard feature has been implemented, but only for select applications, such as with the Microsoft Office clipboard [7], or in specialized tools, such as ClipMate, that are not commonly available to end users [3].

4.1.2 Context-Aware Clipboard

The traditional clipboard model does not support consideration of context. Adding context to copied data could be accomplished with a feature that *extracts context from copied data*. We see a need for such contextual support by the repeat pattern, which had the highest overall usage.

Upon a user copying data onto the clipboard, a context-aware feature would automatically extract useful pieces of that data. Contextual extraction serves then as a disassembler for the copied data to reduce the need to return to the source window. For example, if a user copies an address to paste into a form, a context-aware clipboard could extract the street address, city, state, and zip code each as a separate item. Clearly, this feature would be necessary in conjunction with a multiple-item clipboard. We illustrate this concept in Figure 3 using the repeat pattern as an example. With the current clipboard, the repeat pattern requires five changes in window focus to move three pieces of data from one window to another. A clipboard with a context-aware feature would eliminate many of the unnecessary transitions

between windows *A* and *B*. A further improvement to this model would be to add a feature that *facilitates pasting of multiple items at once*, hence reducing the number of paste operations that must be performed in the destination.

A tool that supports contextual extraction and the ability to paste multiple items at once could transform instances of the repeat pattern into the elementary between pattern, which would reduce the number of changes in window focus to just one. Such contextual support is provided in part by clipboard extensions like Citrine [12] or Entity Quick Click [1]. Citrine extracts context from copied data and uses the contextual information to partition the data. It also allows the user to paste the partitioned data into multiple destination fields with one paste operation, but the user is responsible for teaching Citrine how to map the data to paste. Entity Quick Click allows users to select multiple items with a single click on each item and will extract context from the source application, partitioning the desired text automatically along natural language boundaries, yet it is the responsibility of the destination application to integrate contextual support and to allow for multiple pastes.

4.1.3 Clipboard with Iteration

A clipboard with an iteration feature that *sequentially iterates through multiple items on the clipboard* would allow users to perform the same operation or set of operations on each of several items in a list. A need for such a feature manifests itself in the distribution pattern in which a copy in window *A* results in pastes in windows *B*, *C*, and *D*, and then the user returns to *A*, copies different data, and repeats the pasting process (illustrated in Figure 4). Let us again consider Alex, who is now tasked with transferring contact information for a list of new employees from a spreadsheet into three locations: an address book, a text document, and a web form. To do this, Alex copies data for an employee and pastes it into each of the three destination locations, then returns to the spreadsheet and copies data for the next employee. This is time consuming and repetitive, and could benefit from the concept of an iterator.

We illustrate this concept of a clipboard with iterator in Figure 4. Returning to our example, *A* represents the original spreadsheet containing the employee information, and *B*, *C*, and *D* represent the address book, text document, and web form, respectively. *A'* represents a copy of different data from the same window *A* (another employee's contact information). The destinations *B'*, *C'*, and *D'* represent the pastes of data from *A'* into the same windows *B*, *C*, and *D*. An iterator would reduce the number of changes in window focus by avoiding the need to iterate among *B*, *C*, and *D* depositing the later copies from *A'*. In the proposed clipboard in Figure 4, the dashed arrows between *B* and *B'* represent a user pasting the latter data onto the clipboard, using the iterator, and without changing window focus.

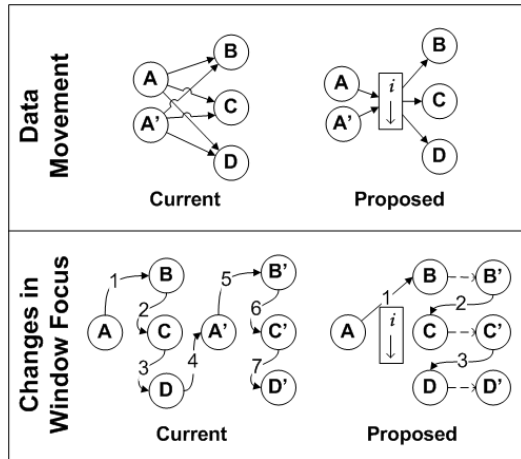


Figure 4. Iterator clipboard, shown with distribution pattern

4.1.4 Window Management

Changing focus between windows is distracting and time-consuming, and it cannot be eliminated completely. We have shown how changes in window focus can be decreased by various automation techniques and motivated the need for a support tool that *reduces the number of changes in window focus*. However, we can also aim to reduce the time it takes to perform this action. This is especially important if the windows overlap, in which case the user must also re-adjust to a new context. To reduce the delays introduced by changes in window focus and context switches during copy and paste activities, advanced window management techniques such as restacking or rolling partially overlapping windows have been shown to be effective [2]. Thus, a feature that *reduces the overhead of managing multiple windows* would be desirable.

4.2. Validation and Format Support

Errors are pervasive in copy and paste activities for professional programmers working within single software applications [6]. Given that end users also perform programming and data aggregation tasks, we conjecture that such errors are likely to occur in end user tasks as well. Unfortunately, end users do not always have the benefit of a compiler or test cases to alert them when errors exist.

It has been observed that programmers often copy and paste data and then modify it in the destination location to adapt it to the new context [6]. Adding *user-defined formatting support to clipboard items* may help reduce the amount of manual formatting users perform when transferring data between two different locations, like transferring a list of websites from a text document into HTML and adding the correct tags. This type of behavior was observed with the repeat pattern and with the distribution pattern, which were the two most popular patterns. Topes is an example of a

data abstraction that could provide such support to the clipboard by allowing users to define how the data should be formatted, but generalizing this solution to data other than small strings is still a challenge [8].

Copy and paste can also lead to context inconsistencies in the paste location. With programming tasks, such inconsistencies can lead to unintended program behavior and errors. A solution to this problem could be to have the *destination applications search for type errors* or have the *destination applications search for context dependencies* that appear when data is pasted. This would be similar to the feature in some IDEs that points out potential compilation errors, or the feature in spreadsheets that identifies inconsistencies between similar or related formulas.

Another error that can occur from copy and paste is inconsistent data between the source and destination resulting from updates in the source location. Consider the task Alex performed in our first example. He compiles monthly reports by copying data from emails, spreadsheets, and websites and pasting it into report fields. If a website data value changes, the report becomes inconsistent and Alex unknowingly presents obsolete information. One solution to this problem is to treat copied data like objects so that all destination locations would reflect updates made to the source data. A feature that *represents pastes as references to copied objects* would provide such a solution. While this might not be necessary for all types of data transfers, it could be practical for reporting data or aggregation tasks. Similar functionality has been provided in part for web applications through Clip, Connect, Clone, which allows users to link data from web pages [4]. Object references could also be enriched by keeping track of the data provenance, as copy and paste actions form links between the source data and its destinations [11]. These links may help users track how data is transferred over time, thus, a feature that *keeps track of the provenance of pasted data* would be useful.

4.3. Overview of the State of the Art

We have now identified and motivated twelve clipboard support features found in the usage data and behavioral patterns we uncovered in our user study. As indicated in this discussion, there exist some tools that address many of these features, but several of the features have not yet been addressed in any existing tools. Table 4 summarizes this set of support features and identifies how existing tools provide some of these features. Some tools are commercial, (Clip-Mate [3] and MS Office [7]), while others are still in a prototype stage (Citrine [12], Quick Click [1], Restacking [2], and Topes [8]).

5 Conclusion

Studying end users and their behavior is no easy task. End users vary greatly in their behavior based on their past

Table 4. Clipboard Support Features

Feature	Citrine	ClipMate	MS Office	Quick Click	Restacking	Topes
Works with all desktop applications	-	+	-	-	+	-
Holds multiple items at once	+	+	+	+	-	-
Extracts context from copied data	+	-	-	+	-	+
Facilitates pasting of multiple items at once	+	-	-	+	-	-
Sequentially iterates through multiple items	-	-	-	-	-	-
Reduces the number of changes in window focus	+	+	+	+	-	-
Reduces overhead of managing multiple windows	-	-	-	-	+	-
User-defined formatting support for clipboard items	-	-	-	-	-	+
Destination apps search for type errors	-	-	-	-	-	+
Destination apps search for context dependencies	-	-	-	-	-	-
Represents pastes as references to copied objects	-	-	-	-	-	-
Keeps track of provenance of pasted data	-	+	-	-	-	-

experiences with technology and the types of tasks they perform. However, in our study of 15 end users, we were able to reveal some of the copy and paste habits of the end users.

We saw that end users are certainly creatures of habit and perform the majority of their copy and paste tasks within just a few applications. We also saw that repeating copy-paste sequences between the same two windows, and distributing data from one source to multiple destination windows, are the most popular behaviors. Furthermore, we witnessed a need for introducing context and iteration to the clipboard to reduce unnecessary changes in window focus and for tracking data provenance that is created through the clipboard. All these findings motivate further study of end user behavior and the development of tools with support features, such as those that we have suggested, to make end users' data transfer tasks faster, more intuitive, and more dependable.

Still, just extending the clipboard may not be the best answer to end users' computational needs. Perhaps it is time to explore alternative and richer metaphors that transform the clipboard from being a mere witness to data transfers, to an intelligent and active participant in end users' attempts to harness the power of their systems.

Acknowledgments

We would like to thank the study participants for their time, Witawas Srisa-an for assisting with recruitment, and the anonymous reviewers of this paper for their helpful comments. This work was supported in part by the EUSES Consortium through NSF-ITR 0324861 and 0325273, and CFDA#84.200A: Graduate Assistance in Areas of National Need (GAANN).

References

- [1] E. A. Bier, E. W. Ishak, and E. Chi. Entity quick click: Rapid text copying based on automatic entity extraction. In *CHI '06 Ext. Abs. Human Factors. Comp. Sys.*, pages 562–567, 2006.
- [2] O. Chapuis and N. Roussel. Copy-and-paste between overlapping windows. In *Conf. Human Factors Comp. Sys.*, pages 201–210, 2007.
- [3] ClipMate 7.3 - The Ultimate Clipboard Extender. <http://www.thornsoft.com/index.htm>, 2008.
- [4] J. Fujima, A. Lunzer, K. Hornbaek, and Y. Tanaka. Clip, connect, clone: combining application elements to build custom interfaces for information access. In *Symp. User Inter. Soft. Tech.*, pages 175–184, 2004.
- [5] M. Kim, L. Bergman, T. Lau, and D. Notkin. An Ethnographic Study of Copy and Paste Programming Practices in OOPL. In *Int'l. Symp. Emp. Soft. Eng.*, pages 83–92, 2004.
- [6] A. J. Ko, H. Aung, and B. A. Myers. Eliciting design requirements for maintenance-oriented IDEs: A detailed study of corrective and perfective maintenance tasks. In *Int'l Conf. Soft. Eng.*, pages 126–135, 2005.
- [7] Copy and paste multiple items by using the Office Clipboard. <http://office.microsoft.com/en-us/word/HA101636021033.aspx>, 2008.
- [8] C. Scaffidi, B. Myers, and M. Shaw. Topes: Reusable abstractions for validating data. In *Int'l Conf. Soft. Eng.*, pages 1–10, 2008.
- [9] C. Scaffidi, M. Shaw, and B. Myers. Estimating the numbers of end users and end user programmers. In *Symp. Visual Langs. Human-Centric Computing*, pages 207–214, 2005.
- [10] C. Scaffidi, M. Shaw, and B. Myers. Games programs play: Obstacles to data reuse. In *WEUSE*, 2006.
- [11] S. Stumpf, E. Fitzhenry, and T. G. Dietterich. The use of provenance in information retrieval. *Work. Princ. Prov.*, Nov. 2007.
- [12] J. Stylos, B. A. Myers, and A. Faulring. Citrine: Providing intelligent copy-and-paste. In *Symp. User Inter. Soft. Tech.*, pages 185–188, 2004.
- [13] G. Wallace, R. Biddle, and E. Tempero. Smarter cut-and-paste for programming text editors. In *Australasian Conf. on User Interface*, pages 56–63, 2001.