Which Interaction Technique Works When?
Floating Palettes, Marking Menus and Toolglasses support different task strategies

Wendy E. Mackay
INRIA
Domaine de Voluceau -Rocquencourt, B.P. 105
78153 Le Chesnay Cedex, FRANCE
wendy.mackay@inria.fr

ABSTRACT
We conducted an experiment that compared three post-WIMP interaction techniques: floating palettes, marking menus and toolglasses, in a real-world Coloured Petri-Net editor, CPN2000. We created six situations in which users performed identical sets of actions with equally-complex nets, but with different cognitive contexts. We found significant differences in performance and preferences across interaction techniques. When a user is in a "copy" context, floating palettes are more efficient. If the user is problem solving, toolglasses or marking menus are preferred. No single interaction technique is clearly superior: each has strengths in different contexts. Since a single application must support different kinds of cognitive tasks, interaction designers should consider integrating multiple interaction techniques, rather than selecting only one.

Categories and Subject Descriptors
H.1.2 [User/Machine Systems]: Human information processing

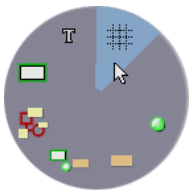
General Terms Design, Experimentation, Human Factors

Keywords
Cognitive Context, Interaction Techniques, Floating Palettes, Marking menus, Toolglasses, Coloured Petri Nets.

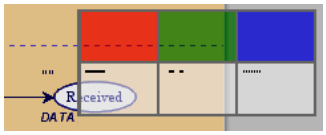
1. INTRODUCTION
Over the past decade, researchers have developed interaction techniques that improve the original desktop metaphor [22] and WIMP (windows, icons, menus, pointing) graphical user interfaces. For example, Kurtenbach [15,16,17] tested contextual *marking menus* and reported speeds of 3.5 times that of traditional pull-down menus on specific tasks. Bier et al. [6] developed two-handed *toolglasses* and Kabash et al. [13] found them to be up to 40% faster than fixed palettes. Unfortunately, despite their promise, we know little about how these interaction techniques compare in real use. Commercial applications rarely use them, partly because they are not supported in standard interface toolkits.

The goals of this study were to compare three post-WIMP interaction techniques in a real-world application and to encourage software designers to consider using them in new applications. We took advantage of a major redesign of a Coloured Petri Net (CPN) [12] editor, Design/CPN, developed by Aarhus University in the 1980's, which uses traditional WIMP interaction techniques. Although not a commercial product, it is used by over 600 industrial and academic organizations world-wide. The new CPN2000 editor, described more fully in [1], incorporates the three interaction techniques: *floating palettes*, *marking menus* and *toolglasses*.

Floating palettes contain sets of command tools represented by buttons. The user clicks on a tool in the palette with the mouse and conceptually holds it in her hand. She can then apply the command to one or more graphical objects on the screen. For example, picking up the "create arc" tool enables the user to create several arcs in a row.



Marking menus are circular, contextual menus that appear when the user clicks the right button of the mouse. They are faster than pull-down menus because it is easier for the human hand to move in a given direction than to reach a target at a given distance. They also provide a smooth transition between novice and expert use, because the menu does not appear if the gesture is executed quickly. The user can move to any on-screen graphical object and apply one or more commands in succession.



Toolglasses, like floating palettes, contain a set of command tools represented by buttons. However, they are semi-transparent and can be moved with the user's non-dominant hand. The user "clicks through" the desired command onto the desired object with the dominant hand. The user can rapidly shift focus between commands and graphical objects on the screen. Note that this interaction technique requires two input devices, in our case, a trackball and a mouse, operated by the user's two hands.

As designers, we expected that one of these interaction techniques would clearly stand out as 'best'. Instead, our field studies and preliminary user tests showed that preferences for particular interaction techniques appeared to vary according to the kind of task and the user's current cognitive context. This paper describes a controlled experiment to identify the circumstances under which users prefer each technique.

Advanced Visual Interfaces AVI'02, May 22-24, 2002, Trento, Italy.

1.1 Tasks reflect different cognitive contexts

As Suchman [23] points out, even goal-oriented tasks are affected by the current work context. Users' actions are situated, influenced by the changing nature of the environment. Green [10] describes the "cognitive dimensions of notations" and classifies six generic activities: incrementation, transcription, modification, exploratory design, searching, and exploratory understanding. He argues against complex task analyses as "lengthy to construct, require specialised experience and in the end do not capture the labile nature of everyday activities." An alternative strategy is to construct scenarios [18, 7] that reflect the subtleties of tasks in context.

We decided to informally test the three interaction techniques above at a retreat for CPN developers. We created prototypes of each and asked 12 CPN developers to use them to work through scenarios based on our field studies of professional CPN developers [11,19]. We had observed that users spent most of their time copying nets already on paper (Green's transcription task) or modifying existing nets to meet new requirements. They also (but rarely) created nets from scratch.

Note that we cannot distinguish these copying and modifying tasks if we examine only the physical actions performed by the user. They differ primarily in the user's cognitive context: his perception of the task and skill level. Copy tasks are almost mechanical, requiring almost no content knowledge, whereas modify tasks require problem-solving skills and a changing focus as the user works through the components of the net.

Consider, for example, a developer editing a Colored Petri Net, or directed graph (Fig. 1). Each net consists of graphical objects with separate shapes and functions. Layout rules are strict, e.g., arcs can only connect places (circles) to transitions (rectangles) or vice versa, but not places to places.

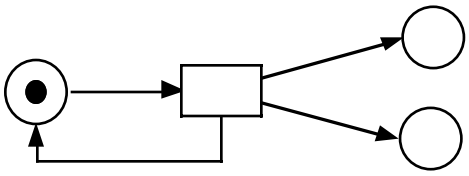


Fig. 1: Simple Petri net with 3 places (circles), one transition (rectangle) and four arcs.

If the user already knows the exact structure and layout of the net to be edited, it is a copying task. The most efficient strategy is usually to create all places first, then all transitions, and then finally join them all with arcs. If the user must make design decisions along the way, it becomes a problem-solving or modify task. Although individual editing commands and the resulting net may be identical, the user's thought processes and the optimal execution strategy differ. Here, the most efficient strategy is to choose a place as a conceptual starting point, then identify the related transitions and link them with arcs, continuing logically until the new net is complete.

What does this imply for testing interaction techniques? Simply that we cannot determine the efficacy of interaction techniques one command at a time. Instead, we must examine the *sequences* of commands performed by the user under different cognitive conditions, if we are to determine which interaction technique is truly more effective.

2. Method

This experiment compares the use of three interaction techniques: floating palettes, marking menus and toolglasses, under two task conditions: copying and modifying nets. We used a 3x2 within-subjects Latin Square design in which each subject was exposed to all six conditions, using a different net for each interaction technique. In each condition, the subject was given a scenario and asked to modify a net using a particular interaction technique accordingly. All scenarios involved creating a new subnet and modifying a set of graphical attributes. The total number of actions for a correct solution was held constant (20-22).

We created one copy scenario and one modify scenario for each of the three nets. We then randomly assigned an interaction technique to each pair of copy and modify scenarios for each net. Each scenario began from the same starting state and subjects were asked to change different, non-overlapping parts of the original net in the copy and modify tasks. The presentation order of nets and interaction techniques was systematically varied and counterbalanced across subjects.

2.1.1 Subjects

Eighteen attendees from the 1999 International Workshop on Practical use of Coloured Petri Nets volunteered to act as subjects. Volunteers signed up for one-hour time slots, in which three subjects were run simultaneously. Subjects all had at least one year of either academic or industrial CPN experience. Ages ranged from 20's to 50's; two were female and sixteen were male. One subject was left-handed. Most subjects had used floating palettes before; none had ever seen or heard of marking menus or toolglasses. Ten subjects were external, eight subjects were from the local CPN group.

2.1.2 Apparatus

The training/debriefing room contained three HP Kayak XW PCs running Windows NT, equipped with a track-ball and a mouse to support two-handed input. The experiment room contained three identical PCs, organized in three separate work areas, with an observer and a subject sitting side-by-side. A video camera was available to tape any of the three work areas.

The software was a working prototype of the CPN 2000 editing tool [1] with the ability to present a single interaction technique (floating palettes, marking menus or toolglasses) or all three together. Many other features were disabled during the experiment, to simplify comparisons. Subjects' keystrokes were logged, providing a timed record of every command used and every object manipulated. This data is sufficient to completely reconstruct the entire session.

2.1.3 Scenarios

We asked a local CPN expert to help us create three test nets that were equivalent in overall difficulty, complexity and in number of graphical components. To ensure their equivalence, we tested the nets and scenarios (over five iterations) with other local CPN experts. The test nets were: Simple Protocol Net (models sending packets over an unreliable network), Distributed Database Net (models an algorithm for ensuring consistency across a number of databases) and Resource Allocation Net (models several processes competing for three different resources). After the experiment, we checked whether subjects treated the nets as equivalent. We ran a one-way analysis of variance of changes, misses, duration and specific editing commands. The lack of statistically-significant differences on any measure ($p<.001$) indicated that the test nets were indeed equivalent for subjects.

We developed six scenarios, two per net, in which users added a new subnet (a set of circles, rectangles and lines) to an existing

net, and changed the graphical attributes (color or line thickness) of specified graphical objects. Half of the scenarios involved copying a net. The subject received a printed net with hand-written annotations and was told to update the on-screen version. A sample copy scenario read:

> *You were unable to go to the design meeting yesterday afternoon, where they made a few changes to the Resource Allocation Net. Your boss gave you her copy of the net, with her notes on the changes to be made. Please use the floating palette to update the net accordingly.*

The remaining scenarios involved modifying a net, requiring a higher level of problem-solving and CPN knowledge. The subject was asked to modify a new version of the same net according to certain criteria. For example: *Modify the Resource Allocation net by creating a new process cycle that competes for the resources R, S and T.* (This is the same as adding a new subnet of circles, rectangles and arcs.) The subject was then asked to ensure that the net met the company's style guidelines, which required changing the color and line thickness of specific graphical objects within the net.

2.2 Procedure

2.2.1 Training

Subjects arrived in a separate training room. After answering various background questions, each subject was given individual, but standardized training, conducted by one trainer. Each subject began the experiment when deemed proficient, defined as able to reliably create, delete and move graphical objects and change graphical attributes, using each of the three interaction techniques (usually after about 10 minutes total).

2.2.2 Experiment

In the experiment room, each subject was assigned to an experimenter who provided standardized explanations of each condition, ensured that the software presented the correct conditions, recorded observations during each condition (with a standard note-taking form), and asked for qualitative impressions immediately after each condition.

Condition	Interaction Technique	Net	Scenario
Training	All	Train	None
1	Marking Menu	2	Copy
2	Marking Menu	2	Modify
3	Floating Palette	3	Copy
4	Floating Palette	3	Modify
5	Toolglass	1	Copy
6	Toolglass	1	Modify
Test		None	None

Table 1: Example of experimental conditions for one subject.

The experiment consisted of six conditions in which each subject was presented with three different nets and asked to perform a *copy* and a *modify* scenario on each, using a different interaction technique for each net. Different subjects experienced different combinations of interaction techniques and nets (Table 1), to control for order effects across nets and interaction techniques. Subjects pressed a button to start each condition; the experimenter pressed a button when the subject had completed the scenario. We needed 18 subjects to present all subjects with all combinations of nets, scenarios and interaction techniques. Each condition has a unique code. Scenarios are labeled A,B, nets are labeled 1,2,3 and interaction techniques are indicated by their initials: FP=floating palette, MM=marking menu and TG=toolglass. For example, a condition with a marking menu, net 2 and the *copy* scenario is: MM-2A.

2.2.3 Debriefing

One experimenter debriefed all subjects, using a standard questionnaire. They were asked both overall preferences among the interaction techniques and also the conditions under which they preferred each technique: creating graphical objects (circular places or rectangular transitions), creating arcs or changing attributes. The experimenter debriefed each subject on the purpose of the experiment, asked for additional suggestions or comments and answered further questions.

2.3 Data Collection

In addition to the background and debriefing information described earlier, we also recorded every user action, e.g., mouse clicks, during every condition. Data was also logged at the command and object level, to facilitate analysis. Every graphical object in each net was assigned a unique identifier. We time-stamped every editing command and recorded the object to which it referred, each object's location, each command's duration and the pauses between commands. This data allowed us to fully reconstruct every condition for every subject and visualize each user's activity patterns (Fig. 2).

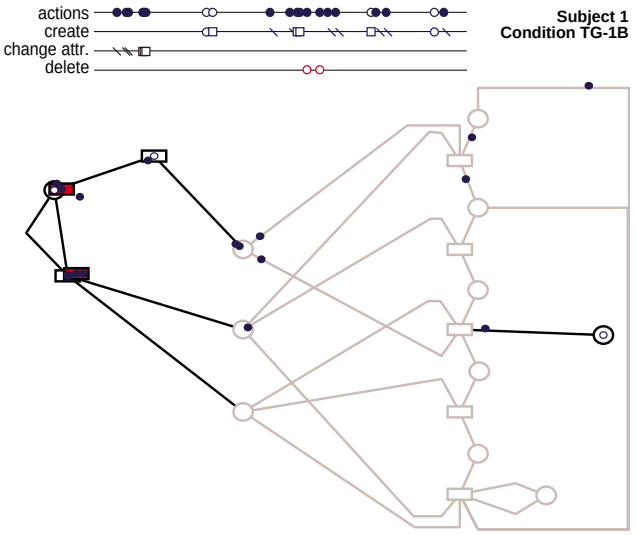


Fig. 2: Data visualization of subject 1, using a toolglass on net 1 in a modify scenario.

Each small dot in the top row indicates a user action. This is a timeline, which makes it easy to identify pauses and clusters of activity. Each row illustrates specific activities: creating objects, changing graphical attributes and deleting objects, with symbols to indicate the specific graphical object. The graphical section displays the original net in light gray and the location and nature of the actions (black dots) and misses (white dots) and edited parts of the net in black. This is interactive: the experimenter can explore the mapping between symbols in the top rows and their location on the actual net.

The user's pattern in Fig. 2 is clearly cyclic. He began by changing several graphical attributes, then started creating a subnet. He created a place, then a transition, then linked them together with an arc. He then created two more transitions, and deleted them. (This is an error: he appears to have mistakenly created the transitions when he was trying to create an arc.) He

then continued with the correct arc, another arc, another transition, two more arcs, a place and a final arc. These actions correspond to the process CPN developers describe when they design or modify a new subnet.

3. Results

The independent variables were: subject (18), task (copy or modify), net (protocol, database, or resource), interaction technique (floating palette, marking menu, or toolglass) and condition order (6). We measured the following dependent variables directly: duration (in milliseconds), total graphical objects before and after each condition, all possible graphical commands, (e.g., create-arc, change-color, move transition), misses, sequences of commands and pauses. We also calculated overall number of actions, number of objects created, deleted or moved, number of uses of each type of command, short (3-6 sec.) and long (>7sec.) pauses and patterns of command use.

We defined *misses* as situations in which the subject tried but was unable to complete a command. For both floating palettes and toolglasses, a miss was defined as attempting to apply a tool without reference to an object, e.g., clicking "delete" on the background. For marking menus, a miss also involved making a gesture that was not recognized, e.g., neither "up" nor "left". Because the algorithm for detecting gestures was overly strict, some subjects had slightly higher levels of misses with marking menus than with other interaction techniques ($F_{2,107}$=45.58, p<.0001). Despite this, misses were not significantly more or less common across task scenarios ($F_{2,107}$=0.32, p>.726) or nets ($F_{1,107}$=.066, p>.797). The following analyses are based on accurate actions, with misses removed, to avoid confounding the data.

Deletes were considered errors, since none were required in any of the scenarios. (Some users "cleaned up" their nets by deleting unwanted objects, others did not.) We found no significant differences across techniques for any delete commands ($F_{2,107}$=. 2.606, p>.0786). The only significant difference we found across task was that users deleted arcs significantly more frequently in the modify scenario ($F_{1,107}$=6.342, p<.0133).

As expected, all three *copy* conditions were shorter, (mean=197sec.) and their durations were not significantly different from one and other. The more difficult *modify* conditions lasted longer, between 5 to 6.5 minutes (mean=345sec.). The difference between these two task conditions was significant ($F_{1,107}$=48.34, p<.0001). The logs show that the time differences are due to pauses, probably as subjects think about the next action, and not to increased use of commands. Total actions were not significantly different across the two task scenarios ($F_{1,107}$=.562, p>.455) or across interaction techniques ($F_{2,107}$=.784, p>.459). The duration of the conditions with respect to interaction technique were not significantly different ($F_{2,107}$=1.153, p>.320), indicating that the techniques were equally efficient for solving the problems.

3.1.1 Interaction patterns

We were particularly interested in the effect of the task and the interaction technique on the subject's activity patterns. Standard graphical user interfaces, with pull-down menus or tool palettes, encourage the user to create all examples of a particular graphical object at the same time, thus reducing the cost of switching between commands. Our field studies (of the old tool) showed that this was the most common practice: creating all places first, then all the transitions, then all the arcs to connect them. This is efficient when the layout is known in advance (as in the copy task), but not when trying to solve a new problem. Working out a problem involves moving through a cycle, creating one type of object, linking it with an arc to another type of object and so on.

We generated two quantitative measures to compare these action sequences. First, we looked at the detailed sequences of actions and identified two categories: *repetitive*, i.e. all objects of one type are created first, followed by all objects of a second type, etc., and *cyclic*, i.e. objects are created in the order expected in a CPN cycle. Fig. 3 shows the percentages of repetitive action sequences for each task type, for each interaction technique. Low percentages indicate a correspondingly high cyclic interaction pattern.

Modify conditions were significantly more likely than *copy* conditions (76% versus 43%) to show CPN cyclic patterns ($F_{1,107}$=13.782, p<.0003). Floating Palette conditions were significantly less likely (25%) to support CPN cyclic patterns than either Marking Menu (72%) or Toolglass (81%) conditions [$F_{2,107}$=17.30, p<0.0001]. The latter were not significantly different from each other. Despite most subjects' prior experience with floating palettes and established repetitive-interaction work styles, only 60% of conditions overall exhibited a cyclic pattern.

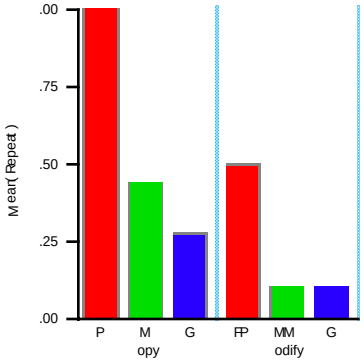


Fig. 3: Percentages of conditions using a repetitive interaction pattern for task-interaction technique combinations.

We also looked for interaction effects among the scenarios and interaction patterns and found that 100% of subjects, when using a floating palette in copy scenario, exhibited the repetitive-interaction pattern. 89% of subjects in the modify conditions, using either toolglasses or marking menus, exhibited the cyclic pattern. In the remaining conditions, the task and technique appear to cancel each other out and were statistically equivalent to each other: the floating palette in the modify condition (50% cyclic) and marking menus (56% cyclic) and toolglasses (72% cyclic) in copy condition.

A second measure of interaction patterns, which included all actions, not just creation commands, calculated the average number of *switches* among commands per condition. A low score indicates a repetitive pattern, e.g., changing all colors at the same time. A high score indicates a cyclic pattern, suggesting that it is easier (because of the interaction technique) or more desirable (because of the cognitive requirements of the task) to switch commands more often.

As before, we found that different tasks and the interaction technique had a strong impact on a user's interaction pattern (Fig. 4). Copy scenarios exhibit significantly more actions

before switching than modify scenarios ($F_{1,107}$=28.23, p<0.0001). Each interaction techniques is significantly different from the others, with floating palettes (mean=3.3 commands before a shift) most likely to encourage repeatedly using the same command and marking menus (mean=2.2) most likely to encourage the opposite pattern, shifting among commands ($F_{2,107}$=10.958, p<0.0001).

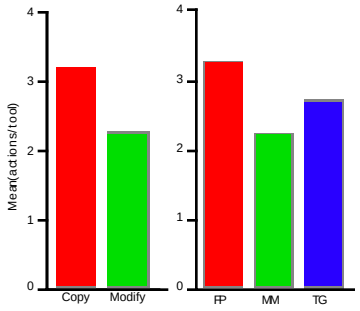


Fig. 4: Average identical actions performed before switching to another command, for tasks and interaction techniques.

The correlation between these two measures is high (Spearman's Rho = .54). The primary difference is the role of marking menus and toolglasses. When we consider only creation actions, as in the first measure, toolglasses are more likely to support cyclic patterns. When all actions, including changing attributes, deleting and moving, are considered marking menus are more likely to exhibit the cyclic pattern.

Cyclic patterns more closely map how CPN developers think about nets. Even though subjects had only about ten minutes exposure to marking menu and toolglasses, and extensive experience with techniques that support repetitive patterns, a majority of subjects found the new interaction techniques as or more efficient than those they habitually used.

3.2 Qualitative Measures

In general, copying tasks should be easier than modify tasks and was evident in the qualitative data. Experimenters noted that one third of the subjects had no problems with any of the six conditions. (These were usually experts, with eight or more years of CPN experience.) Approximately one third found the problems challenging, but ultimately possible. One third of the subjects had difficulty with some or all of the modify scenarios. They had the least CPN experience, 1-3 years.

Seven of the 18 subjects preferred a single interaction technique under most or all situations. Of these, two preferred floating palettes, two preferred marking menus, and three preferred toolglasses. All of those expressing a strong preference for a single interaction technique were junior CPN developers, with one to three years of experience.

Interaction Technique	Create objects	Create arcs	Change attributes
Floating Palette	7	5	3
Marking Menu	7	8	0
Toolglass	4	3	12
No preference	0	2	3

Table 2: Interaction technique preferences by activity (n=18)

Table 2 shows which interaction techniques subjects preferred when performing different specific actions. We distinguished between creating new graphical objects (places and transitions) and creating arcs, because the latter require connecting existing objects on the screen and thus much finer motor control. We also separated changing attributes of graphical objects, such as changing the color or line thickness.

Table 3 presents which interaction techniques subjects preferred for which scenario. At the debriefing, some subjects stated that they preferred floating palettes for thinking tasks, because they would normally think about the net and solve the problem on paper first, then copy the objects onto the screen. (Two subjects did this during the experiment.) These users converted an otherwise "thinking" task into an on-line copying task. Several experimenters commented that subjects who were exposed to floating palettes first tended to use toolglasses as if they were floating palettes and did not switch commands often. Thus, these data likely understate the differences between the copy and modify scenarios and between floating palettes and toolglasses.

Interaction Technique	Copy scenario	Modify scenario
Floating Palette	6	3
Marking Menu	1	4
Toolglass	4	4
No preference	7	7

Table 3: Interaction technique preferences by scenario (n=18)

4. Conclusion

Our original goal was to identify which of a set of promising new interaction techniques (floating palettes, marking menus or toolglasses) would be the "best" for CPN2002, a new Coloured Petri Net editor. We were convinced, both by the literature and our own prototypes, that these techniques were superior to traditional widgets: the problem was choosing one. However, we found that the optimal interaction technique varied according to the task at hand, the user's cognitive context, as well as individual preferences.

In summary, floating palettes are command-oriented: this is the most efficient technique when the user must repeatedly re-issue the same command. Copying tasks, which allow the user to issue similar commands together, are thus best served with floating palettes. Marking menus are object-oriented: they are most efficient when the task involves multiple commands with respect to a single graphical object on the screen. Modification tasks, which involve moving through an existing net on the screen and making sets of changes to each object, are well served by marking menus. Toolglasses involve rapid switching between commands and graphical objects and are useful in tasks that require the user to work through a set of activities that may require a focus on either a new command or a new object.

Different users, either by prior experience or personality, may also have strong individual preferences unrelated to the above. Some users latch on to a particular technique and prefer it for everything, regardless of task or context. Others express an equally strong dislike for a particular interaction technique. In this study, these likes and dislikes were spread evenly across the three interaction techniques.

Interaction technique and cognitive context also affect each other. A floating palettes' tendency to favor repetitive actions can be offset if the user is problem-solving, as in the modify scenario. Similarly, the marking menus' and toolglasses' tendency to favor cyclic actions may be reduced if the user is simply copying. The interaction techniques offer other trade-offs as well. Novice users may benefit when floating palettes

and toolglasses remain visible on the screen, as an on-going reminder of possible actions, but this uses up valuable screen real estate. Marking menus may require active effort on the part of the user to identify the possible alternatives, but provide an easy path from novice (when the menus are displayed) to expert (where gesture alone suffices) use.

Design Implications: Clearly no single interaction technique is optimal in all circumstances for all users. We know from our field studies that users can and do find themselves in both of the cognitive contexts highlighted in this study, on a day-to-day basis. How can a designer support these user requirements?

One strategy is to simply choose one interaction technique and explicitly favor certain types of tasks over others. This is the most common approach and may explain why some software applications designed to support problem-solving tasks are actually better suited for more mechanical copying tasks. This approach is favored by [20], who argues against providing multiple ways of accomplishing the same action and [9], who cautions that performance is not necessarily better if users have a variety of ways to accomplish their goals.

Another strategy would be to create special, intelligent agents that detect what the user is trying to do and automatically provide appropriate interaction techniques. However, it is difficult to see how the system could accurately detect the changes between the copy and modify scenarios used here, since the measurable sets of actions are identical. Unless very accurate, such systems risk reducing the user's productivity, as he repeats commands to recover from system-induced errors.

We propose a third solution, which is to integrate these techniques into a single interface, reducing the cost of access, but not eliminating it. We recognize that integration poses a complex design challenges and describe our own design solution in [2,3]. Our strategy emphasizes providing flexible, "ready-to-hand" access to each technique, by eliminating the concept of a *selection mode* and keeping all objects always accessible. Users can decide which techniques they prefer, under which contexts. We are now working on how to include other interaction techniques, e.g., gestural input [2], zoomable interfaces [4,5], streams [8] and translucent patches [14].

Clearly, we have only scratched the surface of the issues involved in comparing and combining new interaction techniques. Although this study is based on a particular application, we believe the implications extend to a variety of graphical user interfaces. Our data suggest that novel interaction techniques are effective, but their appropriateness depends partially upon the user's task. Researchers attempting to compare interaction techniques cannot simply measure the performance of individual commands. Users use patterns of commands which are strongly influenced by the cognitive context in which the task is being performed. The same set of actions can be very different tasks from the user's perspective, and require different interaction techniques.

Similarly, interface developers must consider how different techniques support these contexts of use. Rather than seeking a single interaction technique for all circumstances, developers should consider how to integrate complementary interaction techniques, in a seamless way, within the interface.

5. Acknowledgments

Special thanks to Michel Beaudouin-Lafon, Anne Vinter Ratzer, Paul Janacek, Katrine Ravn, Michael Lassen, Kasper Lund, Peter Andersen, Mads Jensen, Kjeld Mortensen, Stephanie Munck, Søren Christensen and Kurt Jensen, as well as the Aarhus University CPN group and the participants of the Workshop on Practical Use of Coloured Petri Nets.

6. REFERENCES

[1] Beaudouin-Lafon, M. & Lassen, H.M. (2000) The Architecture and Implementation of CPN2000, A Post-WIMP Graphical Application. In *Proc. of UIST'00*, San Diego, CA, November 2000, CHI Letters 2(2):181-190, ACM Press.

[2] Beaudouin-Lafon, M. (2000) Instrumental Interaction: An Interaction Model for Designing Post-WIMP User Interfaces. In *Proc. of CHI'00* the Hague, Netherlands, p.446-453, ACM Press.

[3] Beaudouin-Lafon, M. & Mackay, W.E. (2000) Reification, Polymorphism and Reuse: Three Principles for Designing Visual Interfaces. In *Proc.of AVI'00* (Palermo, Italy), ACM, p.102-109.

[4] Bederson, B. & Hollan, J. (1994). Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics. In *Proc. of UIST'94*, ACM Press, p.17-26.

[5] Bederson, B., Hollan, J., Druin, A., Stewart, J., Rogers, D., & Proft, D. (1994). Local Tools : an Alternative to Tool Palettes. In *Proc. of UIST'94*, ACM Press. p.169-170.

[6] Bier, E., Stone, M., Pier, K., Buxton, W., & DeRose, T. (1993) Toolglass and magic lenses: The see-through interface. In *Proc.of SIGGRAPH'93*, pp. 73-80.

[7] Carroll, J. (1995) *Scenario-based design. Envisioning work and technology in system development.* NY: Wiley & Sons.

[8] Fertig, S., Freeman, E. & Gelernter, D. (1996). LifeStreams: An Alternative to the Desktop Metaphor. Video in *CHI'96 Adjunct Proceedings*, p. 410-411.

[9] Hertzum, M., & Frøkjær, E. (1996). Browsing and Querying in Online Documentation: A Study of User Interfaces and the Interaction Process. *ACM Transactions on Computer-Human Interaction*, 3(2), 136-161.

[10] Green, T. (2000). Instructions and Descriptions: Some cognitive aspects of programming and similar activities. In *Proc. of AVI'00*, Palermo, Italy, May 2000, ACM. pp. 21-28.

[11] Janecek, P., Ratzer, A. & Mackay, W. (1999) Petri Nets in Use: Redesigning Design CPN. In *Proc. 2nd Workshop Practical Use of Coloured Petri Nets and Design/CPN*. (K.Jensen, Ed.) p. 119-131.

[12] Jensen, K. (1997) Coloured Petri Nets: Basic Concepts (Vol. 1, 1992), Analysis Methods (Vol. 2, 1994), Practical Use (Vol. 3, 1997). *Monographs in Theoretical Computer Science*. Springer-Verlag, 1992-97.

[13] Kabbash, P., Buxton, W. & Sellen, A. (1994) Two-handed input in a compound task. In *Proc. of CHI'94*, ACM Press, p. 417-423.

[14] Kramer, A. (1996). Translucent Patches: Dissolving Windows. In *Proc. of UIST'96*, ACM Press, p. 121-130.

[15] Kurtenbach, G. & Buxton, W. (1994). User Learning and Performance with Marking Menus. In *Proc.of CHI'94*, ACM Press, p.258-264.

[16] Kurtenbach, G., Fitzmaurice, G., Baudel, T. & Buxton. W. (1997). The Design of a GUI Paradigm based on Tablets, Two-hands, and Transparency. In Proc.of CHI'97, ACM Press, p.35-42.

[17] Kurtenbach, G., Fitzmaurice, G.W., Owen, R.N. & Baudel, T. (1999). The Hotbox: efficient access to a large number of menu-items. In *Proc. of CHI'99* ACM Press, p.231-237.

[18] Mackay, W.E. & Bødker, S.(1994) Workshop: Scenario-Based Design. In *CHI'94 Conference Companion*, Boston: ACM Press.

[19] Mackay, W.E., Ratzer, A., & Janecek, P. (2000) Video Artifacts for Design: Bridging the Gap between Abstraction and Detail. In *Proc. DIS 2000*, NY: August 2000, ACM, pp. 72-82.

[20] Raskin, J. (2000) *The Humane Interface*. NY: Addison-Wesley.

[21] Rubine, D. (1991). Specifying Gestures by Example. *Proc. of SIGGRAPH '91*, ACM Press, p 329-337.

[22] Smith, D., Irby, C., Kimball, R., Verplank, B., & Harslem E. (1982). Designing the Star User Interface. *Byte*, 7(4), p.242:282.

[23] Suchman, L. (1987). *Plans and Situated Actions*. Cambridge, England: Cambridge University Press.